МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кваліфікаційна наукова
праця на правах рукопису

ВАСИЛИШИН СВЯТОСЛАВ ІГОРОВИЧ

УДК 004.056.53

ДИСЕРТАЦІЯ

РОЗРОБКА МЕТОДУ ВИКОРИСТАННЯ ПРОГРАМНИХ ПРИМАНОК ЯК ЕЛЕМЕНТІВ ЗАХИСТУ КОМП'ЮТЕРНИХ МЕРЕЖ НА ОСНОВІ ТЕХНОЛОГІЇ BLOCKCHAIN

125 Кібербезпека та захист інформації
(шифр і назва спеціальності)
12 «Інформаційні технологій»
(галузь знань)

Подається на здобуття наукового ступеня доктора філософії
Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ /Василишин Святослав Ігорович/

Науковий керівник (консультант) Опірський Іван Романович, д.т.н, професор

Львів – 2023

АНОТАЦІЯ

Василишин С.І. **Розробка методу використання програмних приманок як елементів захисту комп'ютерних мереж на основі технології Blockchain.** – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 125 «Кібербезпека та захист інформації». – ІКТА Національний університет "Львівська політехніка" України, Львів, 2023.

Дисертаційна робота присвячена вирішенню актуального науково-практичного завдання підвищення ефективності виявлення кіберзлочинів та покращення стійкості захисної системи за рахунок розробки методу використання приманок на основі динамічних атрибутів Blockchain. Це дасть змогу підвищити ефективність захисту комп'ютерних мереж шляхом зменшення навантаження на мережеву інфраструктуру та часу відгуку сервісів для підсилення рівня кіберзахисту інформації в приватних або державних компаніях.

В роботі розроблено метод використання програмних приманок як елементів захисту комп'ютерних мереж на основі технології Blockchain. У сучасному цифровому світі, де кіберзлочинність та кіберзагрози постійно зростають, безпека комп'ютерних мереж стає все більш актуальною темою. У даній роботі автор зосереджується на поєднанні традиційних методів кіберзахисту, таких як програмні приманки, з передовою технологією Blockchain, що дозволяє вивести захист комп'ютерних мереж на новий рівень. Дослідження включає аналіз наявних проблем та рішень, детальне вивчення технології Blockchain та її можливих застосувань у кіберзахисті, а також розробку конкретного методу застосування програмних приманок на основі Blockchain для підвищення ефективності захисту комп'ютерних мереж.

Об'єктом дослідження є приманки з динамічними атрибутами побудовані на основі технології Blockchain з використанням його децентралізованих та стійких до несанкціонованого доступу характеристик для підтримки нормального функціонування пропонованої системи та зберігання атакуючих даних для майбутнього розслідування з метою їх використання для покращення

ефективності захисту.

Предметом дослідження є методи та засоби застосування динамічних атрибутів системи Blockchain для розробки програмних приманок з метою покращення ефективності виявлення кіберзлочинів та забезпечення стійкості захисної системи.

В процесі досліджень використано методи оптимізації, імітаційного та аналітичного моделювання, математичної статистики, об'єктно-орієнтованого програмування, теорії інформації та кодування.

У першому розділі **"Аналіз проблеми використання програмних приманок та огляд наявних рішень"** ретельно вивчається дослідження та публікації, що стосуються актуальної тематики Blockchain та програмних приманок. У цьому розділі проводиться детальний огляд технології програмних приманок, розглядаються їх потенційні можливості, значні переваги, неминучі недоліки, різні рівні взаємодій між компонентамкіи, особливості внутрішнього дизайну та відповідні перешкоди, які можуть виникнути під час їх розгортання. Також у даному розділі звертається увага на актуальні проблеми, пов'язані з використанням приманок та обманок у контексті захисту даних у комп'ютерних мережах. У процесі аналізу проводиться порівняльна оцінка з іншими наявними рішеннями на ринку, зокрема з розвитком та застосуванням технології обману. В результаті дослідження з'ясовано, що, хоча технологія програмних приманок вже вважається застарілою порівняно з іншими сучасними рішеннями на ринку, вона все одно продовжує використовуватися у певних випадках.

У другому розділі **"Аналіз технології Blockchain та дослідження використання її властивостей для розроблення концептуального підходу для використання технології децентралізації в рамках кіберзахисту"** Цей розділ також досліджує можливості застосування Blockchain для захисту даних в різних кібер-областях, зокрема на об'єктах інфраструктури. Розглядаються сфери впливу, в яких дана технологія може розповсюджуватись, та як вона потенційно може використовуватись на об'єктах інфраструктури. Вивчаються можливості використання технології Blockchain в таких актуальних структурах, як урядові та

військові організації, а також перспективи та можливості в комбінуванні з штучним інтелектом, великими даними та іншими передовими технологіями. У рамках розділу аналізується безпековий рівень використання та впровадження технології Blockchain в кіберфізичні структури. На основі цього аналізу виявляються можливі переваги та недоліки використання технології Blockchain для захисту кіберсистем. Проводиться порівняльний аналіз з іншими технологіями та системами, що використовуються для кіберзахисту, і наголошується на сильних та слабких сторонах технології Blockchain в контексті захисту кіберфізичних систем.

У третьому розділі **"Розробка методу використання програмних приманок як елементів захисту комп'ютерних мереж на основі технології Blockchain"** акцентується увага на методі, моделюванні та архітектурі динамічної системи, яка використовує програмні приманки та динамічні властивості Blockchain. У цьому розділі детально вивчається динамічна розподілена система управління, яка використовує динамічні властивості Blockchain та представляється розроблений метод використанні цієї системи. Зокрема, аналізуються параметри, а також представляється динамічна розподілена модель програмних приманок, створена з N хостів та чотирма службами. Модель демонструє, як програмні приманки можуть бути інтегровані з Blockchain-технологією для покращення захисту комп'ютерних мереж. У рамках розділу розглядається архітектура розробленої системи, яка базується на використанні програмних приманок в комбінації з Blockchain-технологією. Досліджується, як різні компоненти системи взаємодіють для забезпечення ефективного захисту комп'ютерних мереж. Крім того, представляється аналіз безпеки розробленої системи за допомогою формального методу Alloy, що дозволяє оцінити надійність та стійкість системи перед різними загрозами.

У розділі 4 **"Дослідження ефективності методу використання програмних приманок побудованих на основі технології Blockchain як елементів захисту"** проводиться дослідження стійкості розробленої системи до різних видів атак, експериментальний аналіз рівня безпеки та оцінка ефективності системи в

порівнянні з існуючими рішеннями. Досліджено рівень безпеки системи та рішення щодо її захисту від трьох видів атак: атак сканування, сніфер та DDoS. Розроблена система використовує RSA 2048-бітовий алгоритм шифрування, який гарантує безпеку каналу зв'язку та запобігає атакам сніферів. Чотири служби періодично відкриваються або закриваються на різних хостах, що забезпечує захист від атак сканування. П'ять розподілених і децентралізованих хостів у системі сприяють ефективному пом'якшенню DDoS-атак. Таким чином, теоретично доведено, що розроблена схема може захиститися від цих атак. Експериментальний аналіз рівня безпеки під час симуляції рішення включає порівняння динамічних та статичних систем, що зазнають обраних атак. Наводяться графіки та кількісні показники результатів, які демонструють відмінності між системами. Проведено порівняльний аналіз ефективності розробленої системи у порівнянні з існуючими рішеннями програмних приманок, які не використовують динамічні атрибути технології Blockchain. Загалом, розроблений прототип виявився стійкішим до зовнішніх атак, ніж статичні аналоги.

Подальша оптимізація прототипу дозволить покращити результат та вдосконалити систему аби вона змогла, як мінімум вдвічі обійти статичні аналоги.

У **висновках** дисертаційної роботи викладено основні результати і рекомендації, які випливають з проведених досліджень, представлено та охарактеризовано кількісні оцінки показників ефективності в умовах використання запропонованих рішень.

У **додатках** до дисертації долучено програмні коди реалізації імітаційних моделей, акти впровадження результатів дисертаційної роботи, а також список наукових праць і апробацій автора за темою дисертації.

Ключові слова: Blockchain, Honeypot, Deception, кібербезпека, кіберпростір, приманки, комп’ютерні мережі.

Список публікацій здобувача:

Наукові праці, в яких опубліковано наукові результати дисертації:

1. Опірський І. Р., Василишин С. І., Сусукайло В. А. Аналіз загроз та безпеки технології NFC при передачі даних для автоматизованої реплікації профілю користувача // Вісник Східно-Українського національного університету імені Володимира Даля. Інформаційна безпека. – 2018. – №3/4 (31/32). – С. 37–44.

2. Опірський І. Р., Сусукайло В. А., Василишин С. І., Луковський Т. І. Розробка методу використання технології NFC для автоматизованої реплікації профілю користувача // Вісник Східно-Українського національного університету імені Володимира Даля. Інформаційна безпека. – 2018. – №3/4 (31/32). – С. 151–157.

3. І.Р. Опірський, С. Василишин, і А. Піскозуб. Аналіз використання програмних приманок як засобу забезпечення інформаційної безпеки. // Кібербезпека: освіта, наука, техніка, – 2020. –вип. 2, вип. 10, – С. 88-97. https://doi.org/10.28925/2663-4023.2020.10.8897

4. Опірський І.Р., С.І. Василишин, В.А. Сусукайло. Розслідування кіберзлочинів за допомогою приманок у хмарному середовищі. Безпека інформації, 27(1). – 2021. – С.13-20. https://doi.org/10.18372/2225-5036.26.15574

5. Susukailo, V., Vasylyshyn, S., Opirskyy, I., Buriachok, V., & Riabchun, O. (2021). Cybercrimes investigation via honeypots in cloud environments. Paper presented at the CEUR Workshop Proceedings, 2923 91-96. (Scopus)

6. Vasylyshyn, S., Opirskyy, I., Shevchenko S. (2021). Honeypot Security Efficiency versus Deception Solution. Paper presented at the CEUR Workshop Proceedings, 3188, 229-236. (Scopus)

7. Vasylyshyn, S., Lakhno, V., Alibiyeva, N., ...Pleskach, V., Lakhno. Information technologies for the synthesis of rule databases of an intelligent lighting control system// M. Journal of Theoretical and Applied Information Technologythis link is disabled, 2022, 100(5), pp. 1340–1353 (Scopus)

8. Опірський І.Р., С.І. Василишин. Перспективи військового застосування технології Blockchain, 28(2). – 2022р. – С.57-66. https://doi.org/10.18372/2225-5036.28.16950

9. Опірський І.Р., С.І. Василишин. Розробка безпеки системи електронного урядування на основі Blockchain, 24(2). – 2022р. – С.58-70. https://doi.org/10.18372/2410-7840.24.16931

10. Vasylyshyn, S., Susukailo, V., Opirskyy, I., Kurii, Y., Tyshyk, I. (2023). A model of decoy system based on dynamic attributes for cybercrime investigation. Eastern-European Journal of Enterprise Technologies, 1 (9 (121)), 6–20. doi: https://doi.org/10.15587/1729-4061.2023.273363 (Scopus)

11. В. Сусукайло С. Василишин, І. Опірський. Дослідження можливостей використання чатботів зі штучним інтелектом для дослідження журналів подій» // НАУ: «Захист інформації». – Том 24, №4 – Київ, 2022р. – С.177-183.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

12. Ivan Opirskyy, Sviatoslav Vasylyshyn, Olexsiy Vavrichen. Game theory method as optimization of Cyber Security Defence // VII Міжнародна науково-технічна конференції “Захист інформації і безпека інформаційних систем” 30-31.05. 2019 р. – Україна, Львів, 2019р. – С.31-32.

13. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitalii Susukailo. Analysis of the use of software baits (honeypots) as a means of ensuring information security // International Workshop on Information Modeling, Zbarazh, Ukraine, 2020, 2, pp. 242–245, 9321925, DOI: 10.1109/CSIT49958.2020.9321925 (Scopus)

14. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitalii Susukailo. Analysis of the attack vectors used by threat actors during pandemic // International Workshop on Information Modeling, Zbarazh, Ukraine, 2020, 2, pp. 261–264, 9321897, DOI: 10.1109/CSIT49958.2020.9321897 (Scopus)

15. Опірський І. Р., Василишин С. І. Аналіз програмних приманок як засобів моніторингу інформації у кіберпросторі // Збірник тез доповідей IV Всеукраїнської науково-практичної конференції молодих учених, студентів і курсантів, ст. 25.

16. Ivan Opirskyy, Sviatoslav Vasylyshyn, Upgrading network efficiency using the honeypot // VIII Міжнародна науково-технічна конференції “Захист інформації і

безпека інформаційних систем” 10-11.11. 2021 р. – Україна, Львів, 2021р. –С.41-42.

17. Опірський І.Р., Василишин С.І., Стан проблеми удосконалення методології використання програмних приманок // “Технічні засоби захисту інформації”, семніар при вченій раді НАН України, Київ, Україна, 2018 р.

18. Опірський І.Р., Василишин С.І., Теорія ігор як засіб для побудови стратегії захисту з використанням програмних приманок. // “Технічні засоби захисту інформації”, семніар при вченій раді НАН України, Київ, Україна, 2020 р.

19. Опірський І.Р., Василишин С.І. Сусукайло В.А., Дослідження вразливості Zerologon // “Технічні засоби захисту інформації”, семніар при вченій раді НАН України, Київ, Україна, 2021 р.

SUMMARY

Vasylyshyn S.I. **Development of a method of using software decoys as elements of protection of computer networks based on Blockchain technology.** – Qualifying scientific work on manuscript rights.

The dissertation for obtaining the scientific degree of Candidate of Technical Sciences (Doctor of Philosophy) in specialty 125 "Cyber Security". - ICTA National University "Lviv Polytechnic" of Ukraine, Lviv, 2023.

The dissertation work is dedicated to solving the topical scientific and practical task of improving the efficiency of detecting cybercrimes and enhancing the resilience of the defense system by developing a method of using decoys based on dynamic Blockchain attributes. This will increase efficiency by reducing the load on the network infrastructure and response time of services to strengthen the level of cyber protection of information in private or state companies.

The work presents a method of using software decoys as elements of computer network protection based on Blockchain technology. In the modern digital world, where cybercrime and cyber threats are constantly increasing, the security of computer networks is becoming an increasingly relevant topic. In this work, the author focuses on combining traditional methods of cyber protection, such as software decoys, with advanced Blockchain technology, which allows taking computer network protection to a new level. The research includes an analysis of existing problems and solutions, a detailed study of Blockchain technology and its possible applications in cyber protection, as well as the development of a specific method of using software decoys based on Blockchain to increase the efficiency of computer network protection.

The object of study is decoys with dynamic attributes built on Blockchain technology, using its decentralized and resistant to unauthorized access characteristics to support the normal functioning of the proposed system and storing attacking data for future investigation.

The subject of research is the methods and means of applying dynamic attributes of the Blockchain system to develop software decoys with the aim of improving the efficiency of detecting cybercrimes and ensuring the resilience of the defense system.

In the course of the research, optimization methods, simulation, and analytical modeling, mathematical statistics, object-oriented programming, information theory, and coding were used.

In the first chapter, **"Analysis of the problem of using software decoys and a review of available solutions"** research and publications related to the current topic of Blockchain and software decoys are carefully examined. In this chapter, a detailed review of software decoy technology is conducted, exploring their potential capabilities, significant advantages, inevitable drawbacks, different levels of interaction between components, features of internal design, and relevant obstacles that may arise during their deployment. Attention is also given to current issues related to the use of decoys and deception in the context of data protection in computer networks. A comparative evaluation with other available solutions on the market is carried out in the analysis, particularly with the development and application of deception technology. As a result of the research, it is found that although software decoy technology is considered outdated compared to other modern solutions on the market, it is still used in some cases.

In the second chapter, **"Analysis of Blockchain technology and the investigation of its properties for developing a conceptual approach to using decentralization technology in the context of cybersecurity"** this chapter also explores the possibilities of applying Blockchain for data protection in various cyber areas, including infrastructure facilities. The areas of influence in which this technology can spread and how it can potentially be used in infrastructure facilities are examined. The possibilities of using Blockchain technology in such relevant structures as government and military organizations, as well as prospects and opportunities in combination with artificial intelligence, big data, and other advanced technologies, are studied. Within the chapter, the security level of using and implementing Blockchain technology in cyber-physical structures is analyzed. Based on this analysis, the possible advantages and disadvantages of using Blockchain technology for cyber system protection are identified. A comparative analysis with other technologies and systems used for

cybersecurity is conducted, emphasizing the strengths and weaknesses of Blockchain technology in the context of protecting cyber-physical systems.

In the third chapter, **"Development of a method for using software decoys as elements of computer network protection based on Blockchain technology"** the focus is on the method, modeling, and architecture of a dynamic system that uses software decoys and dynamic properties of the Blockchain. In this chapter, a detailed study is conducted on the dynamic distributed management system that uses the dynamic properties of the Blockchain and presents the developed method of using this system. In particular, parameters are analyzed, and a dynamic distributed model of software decoys is presented, created with N hosts and four services. The model demonstrates how software decoys can be integrated with Blockchain technology to improve computer network protection. The architecture of the developed system, which is based on the use of software decoys in combination with Blockchain technology, is examined within the chapter. It is investigated how various components of the system interact to ensure effective protection of computer networks. In addition, a security analysis of the developed system is presented using the formal Alloy method, which allows evaluating the reliability and resilience of the system against various threats.

In Chapter 4, **"Investigation of the effectiveness of the method of using software decoys based on Blockchain technology as protection elements"** the resilience of the developed system to various types of attacks, experimental analysis of the security level, and evaluation of the system's effectiveness compared to existing solutions are conducted. The security level of the system and solutions for its protection against three types of attacks: scanning attacks, sniffing, and DDoS, is investigated. The developed system uses the RSA 2048-bit encryption algorithm, which guarantees the security of the communication channel and prevents sniffer attacks. The four services periodically open or close on different hosts, ensuring protection against scanning attacks. Five distributed and decentralized hosts in the system contribute to effective mitigation of DDoS attacks. Thus, it is theoretically proven that the developed scheme can protect against these attacks. Experimental analysis of the security level during the simulation of the solution includes a comparison of dynamic and static systems experiencing the

chosen attacks. Graphs and quantitative indicators of the results are provided, which demonstrate the differences between the systems. A comparative analysis of the effectiveness of the developed system compared to existing software decoy solutions that do not use the dynamic attributes of Blockchain technology is carried out. Overall, the developed prototype proved to be more resistant to external attacks than static counterparts.

Further optimization of the prototype will improve the result and improve the system so that it can bypass static analogs at least twice.

In the conclusions of the dissertation, the main results and recommendations arising from the conducted research are presented and characterized, as well as quantitative assessments of the efficiency indicators in the context of using the proposed solutions.

In the appendices to the dissertation, the software codes for the implementation of simulation models, the acts of implementation of the dissertation results, and the list of scientific works and author's approvals on the topic of the dissertation are included.

Keywords: Blockchain, Honeypot, Deception, cyber security, cyberspace, decoys, computer networks

The list of author’s publications:

Proceedings where basic scientific results of thesis were published:

1. Opirskyy Ivan, Vasylyshyn Sviatoslav, Susukailo Vitaliy. A. Analysis of threats and security of NFC technology for data transmission for automated user profile replication // Bulletin of East Ukrainian National University named after Volodymyr Dal. Information security. - 2018. - No. 3/4 (31/32). - P. 37–44.

2. Opirskyy Ivan, Susukailo Vitaliy, Vasylyshyn Sviatoslav, Lukovskyi T. I. Development of a method for using NFC technology for automated user profile replication // Bulletin of East Ukrainian National University named after Volodymyr Dal. Information security. - 2018. - No. 3/4 (31/32). - P. 151–157.

3. Opirskyy Ivan, Vasylyshyn Sviatoslav, Andrian Piskozub. Analysis of the use of software baits as a means of ensuring information security. // Cybersecurity: education, science, technology, vol. 2, no. 10, pp. 88-97, 2020. https://doi.org/10.28925/2663-4023.2020.10.8897

4. Opirskyy Ivan, Vasylyshyn Sviatoslav, Susukailo Vitaliy. Investigation of cybercrimes using baits in the cloud environment. Information Security, 27(1). - pp.13-20. - 2021. https://doi.org/10.18372/2225-5036.26.15574

5. Susukailo Vitaliy, Vasylyshyn Sviatoslav, Opirskyy Ivan, Buriachok, V., & Riabchun, O. (2021). Cybercrimes investigation via honeypots in cloud environments. Paper presented at the CEUR Workshop Proceedings, 2923 91-96. (Scopus)

6. Vasylyshyn Sviatoslav, Opirskyy Ivan., Shevchenko S. (2021). Honeypot Security Efficiency versus Deception Solution. Paper presented at the CEUR Workshop Proceedings, 3188, 229-236. (Scopus)

7. Information technologies for the synthesis of rule databases of an intelligent lighting control system, Vasylyshyn, S., Lakhno, V., Alibiyeva, N., ...Pleskach, V., Lakhno, M. Journal of Theoretical and Applied Information Technologythis link is disabled, 2022, 100(5), pp. 1340–1353 (Scopus)

8. Opirskyy Ivan., Vasylyshyn Sviatoslav. Prospects for military application of Blockchain technology, 28(2). - pp.57-66. - 2022. https://doi.org/10.18372/2225-5036.28.16950

9. Opirskyy Ivan., Vasylyshyn Sviatoslav. Development of e-government security system based on Blockchain, 24(2). - pp.58-70. - 2022. https://doi.org/10.18372/2410-7840.24.16931

10. Vasylyshyn, S., Susukailo, V., Opirskyy, I., Kurii, Y., Tyshyk, I. (2023). A model of decoy system based on dynamic attributes for cybercrime investigation. Eastern-European Journal of Enterprise Technologies, 1 (9 (121)), 6–20. doi: https://doi.org/10.15587/1729-4061.2023.273363 (Scopus)

11. Susukailo Vitaliy, Vasylyshyn Sviatoslav, Opirskyy Ivan. "research of the possibility of using ai chatbots for event log investigation" // SCIENCE: "Information Protection". - Volume 24, No. 4 - Kyiv, 2022. - P.177-183.

Proceedings that certify an approvement of thesis materials:

12. Ivan Opirskyy, Sviatoslav Vasylyshyn, Olexsiy Vavrichen. Game theory method as optimization of Cyber Security Defence // VII International Scientific-Technical

Conference "Information Protection and Security of Information Systems" 30-31.05. 2019 - Ukraine, Lviv, 2019. - P.31-32.

13. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitalii Susukailo. Analysis of the use of software baits (honeypots) as a means of ensuring information security // International Workshop on Information Modeling, Zbarazh, Ukraine, 2020, 2, pp. 242–245, 9321925, DOI: 10.1109/CSIT49958.2020.9321925 (Scopus)

14. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitalii Susukailo. Analysis of the attack vectors used by threat actors during the pandemic // International Workshop on Information Modeling, Zbarazh, Ukraine, 2020, 2, pp. 261–264, 9321897, DOI: 10.1109/CSIT49958.2020.9321897 (Scopus)

15. Opirskyi I. R., Vasylyshyn S. I. ANALYSIS OF SOFTWARE BAITS AS MEANS OF MONITORING INFORMATION IN CYBERSPACE, Collection of abstracts of IV All-Ukrainian scientific-practical conference of young scientists, students and cadets, p. 25.

16. Ivan Opirskyy, Sviatoslav Vasylyshyn, Upgrading network efficiency using the honeypot // VIII International Scientific-Technical Conference "Information Protection and Security of Information Systems" 10-11.11. 2021 - Ukraine, Lviv, 2021. - P.41-42.

17. Ivan Opirskyy, Vasylyshyn Sviatoslav, The state of the problem of improving the methodology of using software baits // "Technical Means of Information Protection", seminar at the Scientific Council of the National Academy of Sciences of Ukraine, Kyiv, Ukraine, 2018.

18. Ivan Opirskyy, Vasylyshyn Sviatoslav, Game theory as a means of building a defense strategy using software baits. // "Technical Means of Information Protection", seminar at the Scientific Council of the National Academy of Sciences of Ukraine, Kyiv, Ukraine, 2020.

19. Ivan Opirskyy, Vasylyshyn Sviatoslav, Susukailo Vitaliy, Investigation of Zerologon vulnerability // "Technical Means of Information Protection", seminar at the Scientific Council of the National Academy of Sciences of Ukraine, Kyiv, Ukraine, 2021.

ЗМІСТ

ЗМІСТ 15
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 18
ВСТУП 20
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ВИКОРИСТАННЯ ПРОГРАМНИХ ПРИМАНОК ТА ОГЛЯД НАЯВНИХ РІШЕНЬ 30
1.1. Аналіз сучасного стану досліджень та публікацій 30
1.2 . Огляд програмних приманок та їхніх цілей 35
1.2.1. Аналіз переваг та недоліків програмних приманок 39
1.2.2. Рівні взаємодії в програмних приманках 40
1.2.3. Перешкоди для розгортання програмних приманок 42
1.2.4. Внутрішній дизайн програмних приманок 45
1.3. Проблематика використання приманок та обманок для захисту даних у комп'ютерних мережах 47
1.4. Характеристика Blockchain технології 56
1.4.1. Технологія Blockchain 59
1.5. Аналіз використання Blockchain технології 62
Висновки до 1 розділу 67
РОЗДІЛ 2. ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ВЛАСТИВОСТЕЙ BLOCKCHAIN ДЛЯ РОЗРОБЛЕННЯ КОНЦЕПТУАЛЬНОГО ПІДХОДУ З ВИКОРИСТАННЯ ТЕХНОЛОГІЇ ДЕЦЕНТРАЛІЗАЦІЇ В РАМКАХ КІБЕРЗАХИСТУ 69
2.1. Дослідження можливостей використання Blockchain для захисту даних на об'єктах інфраструктури 69
2.2. Безпечна та конфіденційна структура системи електронного урядування на основі Blockchain 73
2.2.1. Розробка методу електронного урядування на основі Blockchain 78
2.2.2. Вузли мережі електронного урядування побудовані з використанням технології Blockchain 80
2.2.3. Голосування делегатів і свідків мережі Blockchain 80
2.2.4. Створення нового вузла мережі Blockchain для електронного урядування 81

2.2.5. Реєстрація користувача в Blockchain мережі для електронного урядування 83

2.2.6. Генерація нового блока мережі Blockchain для електронного урядування 85

2.3. Взаємозв'язок між технологією Blockchain і ШІ, великими даними та іншими передовими технологіями 86

2.4. Перспективи військового застосування технології Blockchain 88

2.4.1. Автономне управління БПЛА в режимі "рою" 92

2.4.2. Військове управління з використанням технології побудованих на основі Blockchain 95

2.4.3. Військова безпека Blockchain мережі 98

2.5. Аналіз безпеки та конфіденційності 99

2.6. Переваги та недоліки використання технології Blockchain для захисту кіберсистем 103

Висновки до розділу 2 106

РОЗДІЛ 3. РОЗРОБКА МЕТОДУ ВИКОРИСТАННЯ ПРОГРАМНИХ ПРИМАНОК ЯК ЕЛЕМЕНТІВ ЗАХИСТУ КОМП'ЮТЕРНИХ МЕРЕЖ НА ОСНОВІ ТЕХНОЛОГІЇ BLOCKCHAIN 108

3.1. Розробка моделі децентралізованого звязку на основі приманок 108

3.1.1. Опис комунікації хостів в побудованій мережі Blockchain 110

3.1.2. Трансформація сервісів під час під'єднання до вузлів 112

3.2. Розробка методу динамічної системи побудованої за допомогою програмних приманок 114

Висновки до розділу 3 121

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДУ ВИКОРИСТАННЯ ПРОГРАМНИХ ПРИМАНОК ПОБУДОВАНИХ НА ОСНОВІ ТЕХНОЛОГІЇ BLOCKCHAIN ЯК ЕЛЕМЕНТІВ ЗАХИСТУ 123

4.1. Аналіз безпекового рівня системи використання програмних приманок на основі розробленої моделі та рішення щодо його захисту 123

4.1.1. Опис проведення експерименту: Сніфер атака 124

4.1.2. Опис проведення експерименту: атака сканування 124

4.1.3. Опис проведення експерименту: DDoS атака 126

4.2. Експерементальний аналіз рівня безпеки під час симуляції рішення 128

4.2.1. Дослідження захисту системи використання програмних приманок на основі розробленої моделі від атаки сніфером 129

4.2.2. Дослідження захисту системи використання програмних приманок на основі розробленої моделі від атаки скануванням 130

4.2.3. Дослідження захисту системи використання програмних приманок на основі розробленої моделі від атаки DDoS 132

4.3. Порівняльний аналіз розробленого динамічного методу з статичними аналогами 142

4.3.1. Порівняння методу використання програмних приманок побудованих з використанням технології Blockchain як елементів захисту з іншими рішеннями 143

Висновки до розділу 4 145

ВИСНОВКИ 147

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 151

ДОДАТОК А. Акти впровадження 161

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Honeypot: системи Honey Pot – це сервери-приманки або системи, налаштовані для збору інформації про зловмисників, які вторгаються в систему.

Puppet: унікальний підхід до автоматизації IT для виявлення, налаштування та керування мережевою інфраструктурою

Віртуальні машини: віртуальна машина — це тип комп'ютерного додатка, який використовується для створення віртуального середовища, яке називають «віртуалізацією».

HonSSH: чудовий інструмент для високорівневої взаємодії honeypot

ОС – операційна система.

VM – Virtual Machine, програма яка дозволяє запускати різні ОС на віртуальній машині та симулювати їхню роботу.

IDS – Intrusion Detection System, система виявлення вторгнень.

IPS – Intrusion Prevention System, система запобігання вторгненням.

DDoS – Distributed Denial of Service, розподілене блокування послуг.

TLS – Transport Layer Security, захист транспортного рівня.

SSH – Secure Shell, протокол безпечного доступу до терміналу.

VPN – Virtual Private Network, віртуальна приватна мережа.

DNS – Domain Name System, система доменних імен.

CPU – Central Processing Unit, центральний процесор.

RAM – Random Access Memory, оперативна пам'ять.

API – Application Programming Interface, інтерфейс програмування додатків.

SIEM – Security Information and Event Management, система управління інформацією та подіями з безпеки.

SSL – Secure Sockets Layer, захищений протокол сокетів.

WAF – Web Application Firewall, мережевий елемент захисту веб-додатків.

DoS – Denial of Service, блокування послуг.

SQL – Structured Query Language, мова запитів до баз даних.

LDAP – Lightweight Directory Access Protocol, протокол доступу до довідникових служб.

DMZ – Demilitarized Zone, зона демілітаризованого доступу.

HTTPS – HyperText Transfer Protocol Secure, захищений протокол передачі гіпертексту.

SFTP – Secure File Transfer Protocol, захищений протокол передачі файлів.

AI – Artificial Intelligence, штучний інтелект.

BYOD – Bring Your Own Device, політика користування особистими пристроями на робочому місці.

CVE – Common Vulnerabilities and Exposures, загальні вразливості та експозиції.

FW – Firewall, мережевий елемент захисту.

GPU – Graphics Processing Unit, графічний процесор.

MFA – Multi-Factor Authentication, багатофакторна аутентифікація.

NAT – Network Address Translation, технологія перетворення мережевих адрес.

OCR – Optical Character Recognition, оптичне розпізнавання символів.

PKI – Public Key Infrastructure, інфраструктура відкритих ключів.

RDP – Remote Desktop Protocol, протокол віддаленого робочого столу.

SCADA – Supervisory Control and Data Acquisition, система нагляду та збору даних.

ВСТУП

Актуальність. Останнім часом зростає інтерес до безпеки та захисту інформації для мережевих систем. Мережні системи містять цінні дані та ресурси, які необхідно захищати від зловмисників. Експерти з безпеки часто використовують honeypots і honeynet для захисту мережевих систем. Honeypot — це видатна технологія, яку експерти з безпеки використовують для виявлення нових методів злому від зловмисників і зловмисників. За словами Шпіцнера [1], засновника проекту Honeynet, «honeypot – це ресурс безпеки, цінність якого полягає в тому, щоб його досліджували, атакували або скомпрометували». Його також можна визначити як «ресурс інформаційної системи, цінність якого полягає у несанкціонованому або незаконному використанні цього ресурсу». Іншими словами, приманка — це дзеркало об'єкту, розміщене в мережі як приманка для заманювання зловмисників. Honeypots, як правило, є віртуальними машинами, призначеними для емуляції реальних машин, дії або створення вигляду запущених повноцінних служб і програм з відкритими портами, які можна знайти на типовій системі або сервері в мережі [2].

Як проактивний захисний механізм, honeypot стає незамінним інструментом для забезпечення безпеки мережі в таких широких додатках, як Інтернет речей (IoT), бездротові сенсорні мережі (WSN), транспортні мережі тощо. Завдяки проактивності та контролю, honeypot може залучити зловмисника до взаємодії з підробленими системними ресурсами, що запобігає атаці на цінні ресурси. У порівнянні з традиційними методами, включаючи, але не обмежуючись, брандмауер і систему виявлення вторгнень (IDS), honeypots повністю скидають пасивність у домені захисту мережі. У зв'язку з цим, приманки привернули широку увагу серед сил кібербезпеки.

Honeypots можна класифікувати на дві категорії з точки зору атрибута варіації: статичні та динамічні honeypots. Статичні honeypots призначені для обману зловмисників, імітуючи деякі характеристики системи. Однак через фіксовану конфігурацію та реакцію вони схильні бути виявлені зловмисниками,

які в подальшому уникають такі надто очевидні пастоки та розпочинають атаку на реальну систему в мережі. Змінна властивість у динамічних програмних приманок покращує слабкість першого. Завдяки змінній конфігурації, псевдосистема (тобто динамічний honeypot) здатна обманювати зловмисників і вивчати їх режими атаки.

Є кілька типових робіт про динамічні програмні приманки. Проблема покращення системи попередження атак та втручання зловмисників досліджувалась вітчизняними та зарубіжними вченими, такими як: Кравець, Степаненко, Мартиненко, Григорчук, Чернів, Лістон, Вірвіліс, Серрано, Ванаутгарден, Акаятта, Сатлгот, Шпітзнер, Хетчер, Хей, Аіндрух, Прокац, Коссаковскі, Рай, Цзян, Морі та інші. Не зважаючи на велику кількість досліджень у цій сфері досі залишаються невирішені проблеми, такі як: централізована система управління безпеки, обчислювальні потужності актуальних захисних систем, людський фактор. Завдяки інтеграції та аналізу даних, зібраних за допомогою пасивних і активних інструментів, можна створити схему динамічної конфігурації та впровадити її в локальну мережу (LAN) для розгортання корпоративної мережі. І справжня, і підроблена системи одночасно діють під час виконання. Потік, ідентифікований як легітимний або агресивний, служить критерієм вимірювання для створення динамічних приманок. Відповідно до навантаження мережі виводиться детальний числовий результат про honeypots і сервери. Результат здійснюється в реальному розгортанні системи або мережі. Крім того, фреймворк керування honeynet можна використовувати для створення підходу динамічної конфігурації. Однак динамічна властивість цих схем honeypot відображається в конфігурації, і розташування все ще є нерухомим. Завдяки технології anti-honeypot зловмисникам легко розпізнати підроблені ресурси та уникнути цих пасток, щоб почати атаку на спаравжні ресурси.

У дисертаційній роботі було використано централізоване управління системою під назвою Puppet для автоматизації конфігурації чотирьох серверів. Для впровадження автоматизованих рішень honeypot була використана віртуальна машина VMware. Дослідження забезпечило цілі з сервісами, такими як Apache

веб-сервер, сервер MYSQL, сервер протоколу передачі файлів (FTP) і сервер простого протоколу передачі пошти (SMTP). Для обробки та індексації журналів використовувався централізований сервер Logstash. Elasticsearch використовувався для зберігання журналів. Kibana використовувався для пошуку та візуалізації журналів.

Спроби зловмисників зламати системи безпеки зростають з кожним днем. Зловмисники використовують такі інструменти, як SubSeven, Nmap і LoftCrack, щоб сканувати, ідентифікувати, досліджувати та проникати в корпоративні системи.

Для запобігання такому несанкціонованому доступу до корпоративних мереж встановлюються брандмауери. Однак брандмауери не можуть запобігти атакам, що надходять з інтранету. Система виявлення вторгнень (IDS) перевіряє мережевий трафік і визначає експлойти та вразливості; він здатний відображати попередження, реєструвати події та електронну пошту адміністраторів можливих атак.

З іншого боку, система запобігання вторгненням намагається запобігти відомим сигнатурам вторгнень і деяким невідомим атакам завдяки знанням про поведінку атак у своїй базі даних.

Однак IDS може щодня генерувати тисячі попереджень про вторгнення, деякі з яких є помилковими. Це ускладнює для IDS виявлення та ідентифікацію реальних загроз і захист активів. Таким чином, для розслідування атак, виявлених і повідомлених IDS, потрібне втручання людини [3].

Побудувати мережу яка базується на основі та засадах технології Blockchain дуже складне завдання та потребує неабияких знань та досвіду в даній сфері. З попереднього твердження випливає й те, що цей процес також не один із найдешевших. Хороші спеціалісти на ринку, які працюють саме в цій сфері дуже дорого обходяться та й до того поки що їх не надто велика кількість. Додаткова складність полягає в тому, що використання готових рішень, наприклад розгалуження від першорівневої мережі, такої як Etherum або Solana, не підходить для реалізації систем захисту інформації, а особливо, якщо говорити про

спеціалізовані системи або об’єкти критичної інфраструктури чи безпеки компанії. Необхідно розробляти свій власний першорівнений Blockchain разом з політиками безпеки, приєднання та від’єднання вузлів (користувачів) до системи, протоколи авторизації та аутентифікації користувачів. В поняття безпеки, коли це стосується важливих та стратегічних об’єктів не повинно бути такого слова, як економія, адже в 80% випадків зловмисник отримує доступ до внутрішньої мережі або успішно проводить DDoS та інші види атак на об’єкти саме через економію на захисті від даної атаки. Решта 20% включає в себе людський фактор, до якого можна зараховувати як і некомпетентність операторів системи (соціальна інженерія), так і її розробників. Однак, поряд з можливостями, які надає Blockchain, необхідно звернути особливу увагу на захист та безпеку учасників мережі, зокрема валідаторів. В сучасному світі різні методи захисту, такі як паролі, біометрія та двофакторна автентифікація через додатки типу Google Authenticator, використовуються на платформах, таких як Binance та Huobi. Проте, на додачу до цих методів, розглядується можливість впровадження NFC-міток для забезпечення ще більш надійного захисту та забезпечення надійності Blockchain-мереж.

В даній дисертаційній роботі розглядається проблема захисту комп'ютерних мереж від кібератак, зокрема використання програмних приманок для залучення та виявлення зловмисників. Однак, існують певні виклики та обмеження, які потрібно вирішувати, щоб забезпечити ефективний захист.

Один з таких викликів - централізація. Для забезпечення властивості транслокації програмних приманок використовується більше одного хоста, а потрібен центральний механізм автоматизації. Однак, традиційний метод використовує центральний хост для підтримки властивості автоматизації. Якщо центральний хост вийде з ладу, то вся система може відмовити.

Другий виклик полягає в складності виявлення програмних приманок. Основна мета механізму динамічної транслокації полягає в тому, щоб приховати реальні ресурси. Однак, завдяки технології anti-honeypot підроблені або справжні ресурси можуть бути розпізнані.

Третій виклик пов'язаний з експертизою. Людина, яка здійснює напад, завжди виходить переможцем. Тому для подальшого розслідування необхідно зафіксувати достовірні докази нападу. Необхідно зафіксувати докази нападу для спеціалістів, які надалі будуть займатися розшуком порушника.

В роботі роздягаються всі переваги та недоліки захисту кібер систем побудованих на технології Blockchain та розробляється концептуальне рішення з використанням програмних приманок побудованих на технології Blockchain, які використовують властивість децентралізації хоста.

Зв’язок роботи з науковими програмами, планами, темами.

Дисертаційні дослідження виконувались у відповідності до наукового напряму кафедри захисту інформації Національного університету «Львівська політехніка» - «Дослідження систем технічного захисту інформації, каналів зв’язку та комп’ютерних мереж, фізичного захисту інформації та криптографії.», в межах кафедральної науково-дослідної роботи: «Розроблення та удосконалення методів і засобів захисту інформації для протидії несанкціонованому доступу в інформаційно-комунікаційних мережах» (шифр ЗІ-7) (№ держреєстрації 0119U101690) (2019р.-2022р.);

- Основні положення та результати дисертаційної роботи впроваджені у навчальний процес кафедри «Захист інформації» Національного університету «Львівська політехніка» при вивченні дисциплін: «Нормативно-правове забезпечення та міжнародні стандарти кібербезпеки» для студентів напрямку підготовки 125 “Кібербезпека”, спеціалізації «Управління інформаційною безпекою»

А також в діяльності підприємства ТОВ "Н-ІКС СПЕЙС".

Мета роботи. *Метою* дисертаційної роботи є підвищення ефективності захисту комп’ютерних мереж та покращення стійкості захисної системи без втрати швидкодії мережевої інфраструктури за рахунок розробки системи приманок на основі динамічних атрибутів блокчейну для підсилення рівня кіберзахисту інформації.

Завдання. Дисертаційна робота присвячена вирішенню актуального науково-

практичного завдання підвищення ефективності виявлення кіберзлочинів та покращення стійкості захисної системи за рахунок розробки системи приманок на основі динамічних атрибутів Blockchain.

Для успішного досягнення мети даної роботи необхідно виконати наступні завдання:

1. Вивчити та проаналізувати наявні рішення та реалізації прогрманих приманок, зокрема їх переваги та недоліки.

2. Дослідити та проаналізувати проблеми та обмеження, пов'язані з використанням програмних приманок у сучасних системах.

3. Проаналізувати властивості Blockchain технології та виявити точки перетину, які можна використати для підсилення технології програмних приманок.

4. Розробити метод використання програмних приманок, який базується на атрибутах системи Blockchain, для забезпечення високої безпеки.

5. Створити модель зовнішніх атак на розроблену систему з метою перевірки її ефективності та виявлення потенційних слабких місць.

6. Провести порівняння реакції запропонованої системи на атаки типу DDoS, сніфер, атака сканування та запити до сервісів із наявними рішеннями, такими як статичні приманки.

Об'єкт дослідження. Об’єктом дослідження є приманки з динамічними атрибутами побудовані на основі технології Blockchain з використанням його децентралізованих та стійких до несанкціонованого доступу характеристик для підтримки нормального функціонування пропонованої системи та зберігання атакуючих даних для майбутнього розслідування з метою їх використання для покращення ефективності захисту.

Предметом дослідження є моделі, методи, алгоритми та засоби застосування динамічних атрибутів системи Blockchain та програмних приманок в мережах зв’язку.

Методи дослідження. В процесі досліджень використано методи оптимізації, імітаційного та аналітичного моделювання, математичної статистики,

об'єктно-орієнтованого програмування, теорії інформації та кодування.

Наукова новизна роботи полягає в тому, що:

1. Вперше розроблена модель динамічної системи активних програмних приманок, що використовують Blockchain технологію. На відміну від відомих дана модель інтегрує децентралізовані та автоматично оновлювані атрибути пасток, що дало змогу підвищити ефективність захисту мережі шляхом зменшення навантаження на мережеву інфраструктуру та часу відгуку сервісів у разі атаки.

2. Набув подальшого розвитку математичний опис обчислення динамічних атрибутів програмних приманок, який, на відміну від відомих враховує динамічні та транс локаційні можливості Blockchain-технології Solana. Це дало змогу змоделювати та оптимізувати розподіл ресурсів мережі за рахунок адаптації до змінних умов, що в результаті сприяло підвищенню ефективності захисту, зокрема забезпеченню швидкого відгуку сервісів під час зовнішніх атак.

3. Вперше, на основі розробленої моделі динамічної системи програмних (активних) приманок, яка використовує Blockchain-технологію для підтримки безпеки, прозорості та адаптивності до зовнішніх атак, за рахунок плаваючих хостів в мережі та математичного апарату обчислення динамічних атрибутів програмних приманок, розроблено метод використання програмних приманок як елементів захисту комп'ютерних мереж на основі технології Blockchain. Цей метод, на відміну від відомих унеможливлює здійснення успішної сніфер атаки за рахунок шифрування, захищає від атаки сканування за рахунок динамічного відкривання та закривання портів, підвищує ефективність захищеності від DDoS атак та збергіє інформацію про атаки на систему на Blockchain-платформі, що забезпечує високий рівень збереження даних та гарантує їхню незмінність.

Практичне значення одержаних результатів полягає у можливості їх безпосереднього застосування для підсилення нявних систем активного та пасивного захисту в корпоративних та державних підприємствах.

1. Розроблена модель динамічної системи активних пасток на основі програмних приманок побудованих на системі Blockchain, яка дала змогу в

залежності від різних ситуацій моделювання (в залежності від атаки) до 54% підвищити пропускну здатність каналу та до 204% підвищити швидкість передачі даних під час проведення зовнішніх атак на систему в порівнянні з статичними аналогами.

2. Удосконалення математичного апарату програмних приманок за рахунок додавання та обчислення динамічних атрибутів програмних приманок дало змогу покращити час відгуку сервісів під час атаки типу DDOS на статичні хости, в межах таких значень – MSQL до 34%, NGNIX до 16%, APACHE до 1%, vsFTPd до 13%.

3. Підвищено ефективність захищеності каналів передачі даних в комп'ютерній мережі за рахунко впровадження розробленого методу динамічної системи активних пасток на основі програмних приманок побудованих на технології Blockchain з RSA 2048 - бітовим алгоритмом шифрування. Система активних пасток не дозволяє декодувати інформацію без відповідного ключа конфіденційності, що забезпечує захист каналу передачі даних та запобігає витоку даних через перехоплення та розшифрування інформації під час її передачі. Експеримент з сніфер атакою на розроблену модель системи показує, ефективність реалізації захисту від перехоплення на основі шифрування.

4. Удосконалення алгоритму визначення та передачі вузлових хостів в системі Blockchain, за рахунок впровадження плаваючих хостів в мережі, підвищило загальну адаптивність мережі реагувати на зовнішні атаки. Цей алгоритм, на відміну від відомих дозволяє системі реагувати на атаки типу сканування та закривати порти доступу реагуючи на зловмисні дії. Результати експерименту під час атаки сканування на відкриті порти дозволили автоматизувати закриття портів, за рахунок зміни основного хоста, що ускладнює збір інформації та можливість доступитись до системи ззовні.

5. Розроблений метод використання програмних приманок, що побудований на основі використання технології Blockchain вимагає більше ресурсів від нападника для здійснення атаки на мережу: потужності комп'ютерів, серверів з яких здійснюється атака а також більше фізичного часу, що збільшує час для

фахівців з кібербезпеки для реагування та контр дії нападу до 45%. Було проведено однакову кількість атак як на централізовану систему, яка використовує програмні приманки так і на динамічну систему, яка побудована на розробленому методі. Найбільшу різницю видно не на всіх сервісах: Apache та Nginx динамічної системи зазнають під час атаки майже однакового з центральним аналогом результатів. Однак сервіси Vsftpd та MySQL вимагають використання значно більших ресурсів від нападника, що показує ефективність розробленого методу у плані захисту комп'ютерної мережі. З тридцяти проведених атак розроблена модель успішно заблокувала 50% в той час як централізована всього 13%, що показує покращення захисних можливостей розробленої моделі. Захист децентралізованої моделі мережі під час DDoS атаки вищий на 14%. Захист децентралізованої моделі мережі під час атаки сканування вищий на 44%, майже в два рази. Захист децентралізованої моделі мережі під час сніфер атаки вищий на 37%. Загальний захист комп'ютерної мережі побудованої з використанням прогрманих приманок на основі технології Blockchain вищий на 37% в порівнянні з централізованим аналогом, що є підвищенням глобального рівня захисту комп'ютерної мережі в півтора рази.

Наукові та практичні результати виконаних досліджень використані у навчальному процесі кафедри телекомунікацій Національного університету «Львівська політехніка», зокрема для студентів спеціальності 125 «Кібербезпека» в курсі лекцій з дисципліни «Нормативно-правове забезпечення та міжнародні стандарти кібербезпеки».

Основні результати дисертаційної роботи використано і впроваджено з метою покращення захищеності комп'ютерної мережі та систем в компанії ТОВ «Н-ІКС СПЕЙС», що підтверджено актами впровадження.

Особистий внесок. Основні наукові результати дисертаційної роботи отримано автором самостійно. У працях, опублікованих у співавторстві, внесок Василишина С.І. є вирішальним, зокрема авторові належать (нумерація згідно Додатку В): у роботах [3, 4, 8] – розроблення архітектури динамічної системи honeypot на основі Blockchain-технології, [11, 14, 15] – аналіз та дослідження

наявних рішень побудови мереж з використанням програмних приманок [1, 16, 17] – розроблена модель динамічної системи активних пасток (honeypots) на основі програмних приманок, що використовують Blockchain технологію, [5, 6, 7] – вдосконалення математичного апарату на якому базується технологія програмних приманок та обманок, [18, 19, 20] – моделювання та дослідження використання вдосконалених програмних приманок в області кібербезпеки.

Апробація результатів. Основні наукові результати і положення дисертації представлені, доповідались та обговорені на 8-ми міжнародних і державних науково-технічних конференціях та наукових семінарах: VII Міжнародна науково-технічна конференції “Захист інформації і безпека інформаційних систем” (м. Львів, 2019, 2021 рр.), IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT) (Zbarazh, 2020 р.), IV Всеукраїнської науково-практичної конференції молодих учених, студентів і курсантів (м. Київ, 2020 р.), семніар при вченій раді НАН України “ Технічні засоби захисту інформації ” (м. Львів, 2018, 2020, 2021 рр.). Крім цього дисертаційна робота у повному обсязі представлена на наукових семінарах кафери Захисту інформації Націоланльного університету “Львівська політехніка”.

Публікації. Основні результати дослідження викладено у дев’ятнадцяти наукових публікаціях, а саме: у одинадцяти статтях (із них сім – у фахових наукових виданнях України та чотири – у періодичних виданнях зарубіжної держави) і восьми тезах виступів на науково-практичних заходах.

Структура та обсяг дисертації. Дисертація складається з вступу, чотирьох розділів, що охоплюють 16 підрозділів, висновків, списку використаних джерел і додатків. Загальний обсяг дисертації становить 191 сторінки, з яких 150 – основний текст, 10 – список використаних джерел (101 найменування), 31 – додатки.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМИ ВИКОРИСТАННЯ ПРОГРАМНИХ ПРИМАНОК ТА ОГЛЯД НАЯВНИХ РІШЕНЬ

1.1. Аналіз сучасного стану досліджень та публікацій

Протягом останніх кількох років було запропоновано багато різних варіантів використання програмних приманок. Деякі з них використовуються, щоб витрачати час хакерів [4], інші для зменшення активності спаму [5] або для обману зловмисників [6], а деякі для аналізу кроків хакерського вторгнення [7]. Протягом кількох років спільнота безпеки використовувала honeypots для аналізу різних методів, які застосовували зловмисники.

Шпітзнер представив два типи honeypots: низькорівневі пригоди взаємодії – це комп'ютерне програмне забезпечення, яке емулює операційні системи та послуги, які зазвичай застосовуються у виробничому середовищі в організації, і високорівневі пригоди, які передбачають розгортання реальних операційних систем на реальні або віртуальні машини – це також використовується в дослідженнях безпеки. Крім того, Шпітзнер визначив дві критичні вимоги архітектури honeynet: (1) контроль даних зменшує ризик, гарантуючи, що коли зловмисник проникає в системи honeynet, ці скомпрометовані системи не можуть бути використані для атаки або шкоди іншим системам, і (2) захоплення даних гарантує, що експерти з безпеки можуть виявляти та фіксувати всі дії, які виконує зловмисник, навіть якщо вони зашифровані. У цьому дослідженні використовувалися маніпулятори високої взаємодії для збору даних із реальних систем.

Собесто та ін. представив DarkNOC, управління та інструмент моніторингу для складних honeynet [8]. Він складається з різних видів Honeypots, а також інші пристрої збору даних. Вони спроектували рішення, яке здатне обробляти велику кількість шкідливого трафіку, отриманого великою мережею Honeynet, і ефективно надавати зручний веб-інтерфейс для показу потенційних скомпрометованих хостів адміністраторам мережевої безпеки, а також забезпечує загальний стан безпеки мережі. У цьому дослідженні Kibana реалізував зручний

веб-інтерфейс, щоб візуально показати атаки з високою частотою. Діттріх зазначив, що розподілена мережа програмних приманок може збільшуватися в розмірах [9]. Один honeypot не може ефективно контролювати та керувати атаками.

Аналіз даних про атаки, що походять від великої кількості індивідуальних honeypots в мережі, є складним. Діттріх рекомендував використовувати Manuka, інструмент баз даних із переднім і заднім клієнтом для вирішення цієї складної проблеми. База даних містить атрибути систем, які доповнюють пошук типу операційної системи, версії та встановлених служб. Він заявив, що інструмент Manuka може взяти лише одну людину для встановлення honeypots, завантаження його в базу даних і швидкого розгортання в розподіленій мережі. Аналогічно, у цьому дослідженні використовувався інструмент керування автоматизацією Puppet з відкритим вихідним кодом для розгортання серверів і служб на honeypots без ручного втручання.

Вайлер запропонував систему, яку можуть застосовувати великі організації для захисту від розподілених атак відмови в обслуговуванні. Його дослідження спиралося на технологію honeypot, яка має дві переваги.

Вайлер включає в себе мережу демілітаризованої зони, яка реалізує такі послуги, як веб, пошта, ftp і DNS для доступу через зовнішні мережі. Локальна внутрішня мережа (LAN) організації знаходиться в іншій зоні, захищеній брандмауером. Потім інфраструктура представила нову систему: honeypot, яка буде імітувати внутрішню мережу та залучати DDoS-зловмисників. Крім того, Вайлер створив внутрішні системи в організації, щоб вони діяли як приправа. «Наприклад, якщо виявлено скомпрометовані пакети зловмисника на веб-сервер корпорації, пакети надходять у honeypot для обробки. Відповідь, яку отримує зловмисник, не можна відрізнити від справжньої відповіді веб-сервера» [10].

Наведений вище дизайн вирішив три проблеми, а саме: атаки можна виявити, пакети атаки можуть бути активно спрямовані на honeypot, а honeypot здатний імітувати мережеву інфраструктуру організації, принаймні частини, відомі зловмиснику. Вілсон, Маймон, Собесто та Цукер досліджували вплив

банера спостереження в атакуваній комп’ютерній системі, який знижував ймовірність введення команд у систему під час триваліших перших інцидентів з проникненням в систему. У дослідженні також зазначено ймовірність кількості команд, що вводяться під час наступних інцидентів проникнення в систему, в тій же системі, визначається наявністю банера спостереження та тим, чи були команди введені в результаті попередніх інцидентів з проникненням [11].

Результати Вілсона та інших показують, що середня кількість інцидентів проникнення в систему на один комп’ютер становить 4,44 і 4,48 для комп’ютерів без банерів і банерів спостереження відповідно [12]. Це не статистично значуща різниця; проте присутність вплинула на серйозність інцидентів з проникненням на територію. У дослідженні також зазначено, що зловмисники, які отримують банер спостереження, які провели в системі щонайменше 50 секунд, мають на 8% менше шансів вводити команди в систему, ніж відповідні зловмисники, які не отримують банер спостереження. З тих, хто раніше не вводив команди в систему, 38% тих, хто має банер спостереження, і 47% тих, хто не вводив команди, вводили команди під час другого інциденту з проникненням. Більше того, з тих, хто вводить команди в систему, 67% тих, хто має банер спостереження, і 63% тих, хто не має банерів спостереження, ввели команди під час другого інциденту з проникненням.

Дослідження походить із рандомізованого контрольованого дослідження, проведеного у великому державному університеті Сполучених Штатів [13]. Протягом 7-місячного експериментального періоду було розгорнуто 660 цільових комп’ютерних систем після того, як їх зламали порушники системи. Ці системи викликали 2942 інциденти з проникненням в систему (Вілсон та інші, 2015). Знову ж таки, вони сказали, що кожна з цих систем була випадковим чином призначена для відображення банера відеоспостереження (n=324) або не показу спостереження (n=336) при кожному вході до відповідної системи. Банер відеоспостереження зображений ліворуч саме так, як його бачили зловмисники, з повідомленням: «Ця система знаходиться під безперервним наглядом. Вся активність користувачів контролюється та записується» [14].

Стокман, Рейн та Хейле використовували систему Honeynet з відкритим вихідним кодом для вивчення впливу системного банерного повідомлення на хакерів. Дослідження проводилося протягом 25 днів, збираючи дані про майже 200 000 подій [15].

За даними Стокмана та інших, було дозволено 510 входу через підробку паролів, і з 510 входу 280 отримали банер з попередженням, а 230 не отримали банер взагалі [16]. Середня тривалість спроб для тих, хто має банер-попередження, становила 15,29 секунд, а для тих, хто не має, 23,45 секунди. Середня тривалість тих, хто мав банер із попередженням, становила 9 секунд, а тих, хто не мав, — 11 секунд. Крім того, зловмисники нічого не робили під час входу в систему через те, що система записувала лише кілька команд. Однак вони заявили, що дослідження не дозволило зловмисникам увійти як root, і це може призвести до зниження рівня активності [17].

У роботах Стокмана та інших використано ручне втручання при налаштуванні honeynet мережі. Однак у цьому дослідженні досліджувалися способи створення цілей, які представляють більший інтерес для хакерів для атаки на такі служби, як веб-сервери або сервери баз даних, які, здається, є ресурсами, що представляють реальну цінність для зловмисників [18].

Дьорінг запропонував п'ять тестових випадків, у рамках яких honeypot можна розгорнути по-різному в окремому середовищі, і пояснює їх окремі атрибути. Згідно з його дослідженням, «кількість атак, що відбуваються в захищеному середовищі, менша, ніж кількість атак із незахищеного середовища, принаймні, вони повинні. Тому після порівняння результатів потрібно зосередитися на навколишньому середовищі» [19].

Результати дослідження Дьорінг надали практичний підхід до розгортання honeypot. Він обговорив чотири фази: аналіз, розробка, реалізація та фаза висновку. За його словами, перший етап розробки починається зі збору знань про Honeypots та визначення вимог.

Хоке і Бікас представили систему виявлення вторгнень (IDS), яка використовує генетичний алгоритм (GA) для ефективного виявлення різних типів мережевих

атак. Їх підхід використовував теорію еволюції до еволюції інформації, щоб відфільтрувати дані про трафік і таким чином зменшити складність [20]. Вони використовували випадково згенеровану популяцію хромосом, які представляють усі можливі рішення проблеми, які вважаються кандидатами. З кожної хромосоми різні позиції кодуються у вигляді бітів, символів або чисел.

Крім того, варто зазначити, що деякі схеми honeypot є динамічними. Динамічна властивість в основному відображається на конфігурації та розгортанні [21]. Куватлі пропонує динамічну схему honeypot і використовує Nmap, P0f і Snort для активного виявлення та пасивного розпізнавання відбитків пальців [22]. Honeyd і деякі дуже інтерактивні honeypots використовуються для моделювання мережі та перенаправлення мережевого потоку відповідно. Динамічний механізм honeypot взаємодіє із згаданими вище модулями, динамічно конфігуруючи Honeyd та надаючи інтерфейс, який можна конфігурувати. Хасан та ін. динамічно налаштовує honeypots, щоб імітувати реальну промислову мережу в режимі реального часу (тобто honeypot є фіктивом реальної системи) і виділяє невикористані IP-адреси кластеру Honeyd [23].

Саеді та ін. вивчає динамічне керування приманками. Відповідно до даних, зібраних з маршрутизаторів, брандмауерів, IDS і honeypot, конфігурація honeypot динамічно коригується для адаптації до мережевого середовища [24]. Фан та ін. поєднує високоінтерактивний honeypot з низькоінтерактивним. Адаптивна схема honeynet реалізована шляхом моделювання деяких операційних систем. Ключовим модулем цієї схеми є шлюз Honeybrid, який містить частини прийняття рішень і перенаправлення. Перший використовується для захоплення та передачі певного мережевого трафіку в Honeyd. Останній має на меті перенаправити потік Honeyd на високоінтерактивна програмна приманка. У наведених вище роботах йдеться про динамічну схему конфігурації honeypot [25]. Є деякі роботи щодо динамічного розгортання honeypot. Хечер і Хей пропонують схему автоматизації розгортання honeypot [26]. Для моніторингу мережі використовуються активні та пасивні технології виявлення потоків мережі. Інформація про конфігурацію користувача зберігається в базі даних, яка може служити критерієм класифікації

для створення нової мережі honeynet, обмеження пропускної здатності та цільового діапазону IP мережі. Динамічна схема Honeypot Honeyvers, заснована на машинному навчанні, запропонована Фраунхольц та ін. Мережне середовище сканується, і обладнання класифікується, щоб визначити точну кількість honeypots, таким чином автоматично генеруючи інформацію про конфігурацію та подальше розгортання honeypots [27]. Щоб вирішити проблему, викликану нерівномірним розгортанням honeynet, Фан та ін. висуває архітектуру керування кількома віртуальними мережами, яка генерує конкретну інформацію honeynet на основі різних запитів. Індивідуальна honeynet автоматично розгортається набором інструментів [28].

Ці динамічні схеми honeypot роблять вигляд, що вписуються в мережеве середовище самоадаптивно і зосереджені на шахрайстві зловмисників. Однак розташування цих програмних приманок, здається, фіксується після визначення конфігурації або інформації про розгортання. З розвитком технології anti-honeypot всі ці проекти, швидше за все, уже будуть знайдені та вираховані. Через властивість перетворення розташування в пропонованій схемі даної роботи ці динамічні конфігурації honeypots відрізняються від інших [29]. В пропонованій схемі, навіть якщо зловмисники виявляють honeypot, вони не можуть знайти реальні ноди та користувачів.

1.2. Огляд програмних приманок та їхніх цілей

Перш за все, honeypot – це комп’ютерна система. У ньому є файли, каталоги, як у справжньому комп’ютері. Однак мета комп’ютера – залучити хакерів, щоб вони потрапили в нього, щоб спостерігати та стежити за їхньою поведінкою. Тому ми можемо визначити це як фальшиву систему, яка виглядає як справжня система. Вони відрізняються від інших систем безпеки, оскільки вони не тільки знаходять одне рішення певної проблеми, але також мають право застосовувати різноманітні проблеми безпеки та знайти кілька підходів до них. Наприклад, їх можна використовувати для реєстрації шкідливих дій у скомпрометованій системі, їх також можна використовувати для вивчення нових загроз для користувачів і створення ідей, як позбутися цих проблем. Можна

розділити honeypots відповідно до їх цілей та рівня взаємодії. Якщо ми подивимося на цілі приманок, то можна побачити, що є два типи програмних приманок: дослідницькі та виробничі [30].

Тенденції зростання користувачів технології програмних приманок.

2010-2012: Помірне зростання, оскільки Honeypot/Deception технології все ще знаходяться на ранніх етапах розвитку і більш активно використовуються дослідниками безпеки та організаціями з високим рівнем знань у сфері кібербезпеки [31].

2013-2016: Швидке зростання, в результаті більш широкого визнання Honeypot/Deception технологій та їхньої ефективності в боротьбі з кіберзлочинністю. Розвиток відкритих джерел та комерційних продуктів також сприяє поширенню цих технологій.

2017-2023: Стабільне зростання з підвищеною швидкістю, оскільки більше компаній та організацій інтегрують Honeypot/Deception технології в свої системи кібербезпеки. Спеціалісти з кібербезпеки та компанії все більше використовують ці рішення для проактивного виявлення та засікання загроз.

Схематичну діаграму тенденції зростання можна побачити на рис. 1.1.

Рис. 1.1. Схематична діаграма тенденції використання програмних

приманок користувачами.

Дослідницькі приманки в основному використовуються військовими, дослідницькими та державними організаціями. Вони збирають величезну кількість інформації. Їхня мета – виявити нові загрози та дізнатися більше про мотиви та прийоми Blackhat. Мета полягає в тому, щоб навчитися краще захищати систему, вони не приносять ніякої прямої цінності безпеці організації [32].

Виробничі honeypots використовуються для захисту компанії від атак, вони впроваджуються всередині виробничої мережі для підвищення загальної безпеки. Вони збирають обмежену кількість інформації, в основному використовуються приманки з низьким рівнем взаємодії. Таким чином, адміністратор безпеки уважно стежить за переміщеннями хакера і намагається знизити ризики, які можуть виникнути від нього для компанії. На цьому етапі ми спробуємо обговорити та з’ясувати ризики використання виробничих програмних приманок. Оскільки під час тестування безпеки систем, що існують в організації, можуть статися несподівані дії, наприклад, неправомірне використання інших систем із використанням функцій honeypot. Якщо адміністратор мережі не знає про цю проблему, вони створюють великі проблеми для організації [33].

Групи програмних приманок можна поділити на: запобігання, виявлення та реагування, які схематично зображені на рис. 1.2.

Рис. 1.2. Поділ програмних приманок на групи.

- Запобігання — це перше, що слід враховувати в нашій моделі безпеки. Як визначення, це означає запобігти хакерам зламати систему. Тому ми намагатимемося не дати їм доступу до системи. Є багато способів зробити це в безпеці. Можна використовувати брандмауер для управління мережевим трафіком і встановити деякі правила, щоб блокувати або дозволяти його. Використання методів аутентифікації, цифрових сертифікатів або наявність надійних паролів є найпоширенішими і добре відомими методами захисту. Існують також алгоритми шифрування, які шифрують дані [34]. Це хороший спосіб використовувати його, оскільки він шифрує повідомлення і робить їх неможливими для читання. Зв'язок між застосуванням профілактики та програмних приманок можна пояснити наступним чином. Якщо хакер розуміє, що компанія, яку він намагається зламати, використовує honeypots, і він усвідомлює сьогоднішні проблеми безпеки, це змусить їх задуматися про це. Для хакера це буде заплутано і страшно. Навіть якщо компанія використовує методи, які ми обговорювали в першому абзаці, щоб залишатися в безпеці, все одно добре мати honeypot в організації, оскільки питання безпеки стурбовані та вирішуються професійно. Оскільки безпека дуже важлива, завжди добре бути свідомим. Немає терпимості, коли є проблема, вона може завдати великої шкоди будь-якій компанії. Оскільки кожна компанія має конфіденційні та важливі дані, необхідно захистити дані від зловмисників [35].

- Виявлення – це акт виявлення будь-якої шкідливої діяльності в системі. Припускається, що профілактика так чи інакше не спрацювала, систему зламав хакер. Є кілька способів виявлення цих атак. Добре відомим рішенням для виявлення є Network Intrusion Detection Systems. Ця технологія допоможе користувачам дізнатися, чи зламана мережа, але вона не завадить хакерам атакувати систему. Для компаній такі системи виявлення дорогі. На цьому етапі маніпулятори є цінними для контролю активності [36].

- Реагування. На даному етапі ми впевнені, що відбувся напад, і буде відповідь на це. Ось тут і починається розслідування. Коли хакер скомпрометує систему, він залишає сліди. За допомогою відповідних інструментів можливо обробляти дані таким чином, щоб мати певні підказки про те, що сталося з

системою. Можна переглянути файли журналів і спробувати дослідити, що сталося.

1.2.1. Аналіз переваг та недоліків програмних приманок

На ринку доступно багато рішень безпеки. Кожен може переглянути різноманітні варіанти в Інтернеті та знайти найбільш підходяще рішення для своїх потреб. Переваги та недоліки відображені у таблиці 1.1.

Таблиця 1.1

Переваги та недоліки використання програмних приманок

Переваги	Недоліки
Honeypots можуть фіксувати атаки та надавати інформацію про тип атаки, а за потреби, завдяки журналам, можна побачити додаткову інформацію про напад. Переглядаючи їх, можна побачити нові атаки та створити нові рішення безпеки. Додаткові перевірки можна отримати, подивившись на тип зловмисної поведінки. Це допомагає зрозуміти більше нападів, які можуть статися. Honeypots не є громіздкими з точки зору збору даних.	Захоплення даних можливе лише тоді, коли хакер активно атакує систему. Якщо він не атакує систему, зловити інформацію неможливо. Якщо в іншій системі відбувається атака, honeypot не зможе її ідентифікувати. Отже, атаки не на систему honeypot можуть пошкодити інші системи та спричинити великі проблеми.
Вони мають справу лише з вхідним шкідливим трафіком. Тому інформації, яка була зловлена, не так багато, як весь трафік. Зосередження лише на шкідливому трафікі значно полегшує розслідування. Тому це робить приманки дуже корисними. Для єдиного шкідливого трафіку немає	Є недолік відбитків у honeypots. Досвідченому хакеру легко зрозуміти, атакує він систему honeypot чи реальну систему. Відбитки дозволяють нам розрізняти ці два. Це не бажаний результат.

потреби у величезному сховищі даних.	
Немає необхідності в обслуговуванні нових технологій. Будь-який комп'ютер можна використовувати як систему honeypot. Таким чином, створення такої системи не вимагає додаткових витрат. Їх легко зрозуміти, налаштувати та встановити. Вони не мають складних алгоритмів. Немає необхідності оновлювати чи змінювати деякі речі.	Honeypot можна використовувати як зомбі, щоб дістатися до інших систем і скомпрометувати їх. Це може бути дуже небезпечним.
Оскільки honeypots може захоплювати будь-що шкідливе, він також може захоплювати нові інструменти для виявлення атак. Це дає більше ідей і глибини теми, що доводить, що можна виявити різні точки зору та застосувати їх для рішень безпеки.	

1.2.2. Рівні взаємодії в програмних приманках

Оскільки honeypots класифіковано відповідно до їхніх цілей, тепер настав час детальніше розглянути рівні взаємодії. Рівень взаємодії означає, наскільки хакер може взаємодіяти з системою [37]. Більша кількість даних, яку хотілося б зібрати, вимагає більшого рівня взаємодії. Більший рівень взаємодії також приносить більше ризиків для безпеки мережі. Існує три категорії рівнів взаємодій у honeypots. Їх називають низькою взаємодією, середньою взаємодією та високою взаємодією, які відображені на рис. 1.3.

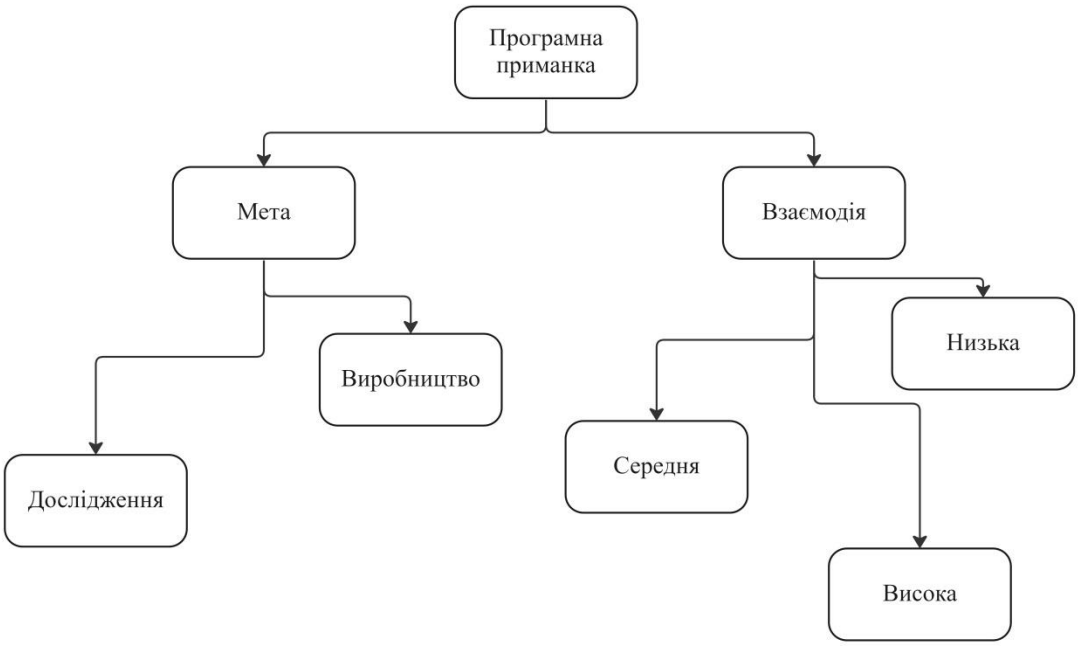

Рис. 1.3. Рівні взаємодій системи програмних приманок

За допомогою приманок з низькою взаємодією можна отримати найменшу кількість даних порівняно з іншими системами honeypot. Вони обмежені, тому ризик, на який потрапив зловмисник, також не є пропорційним. Перш за все, немає операційної системи, з якою можна мати справу. Їх можна використовувати для виявлення нових хробаків або вірусів і аналізу трафіку, що проходить через мережу. Прилади низького рівня взаємодії легко налаштувати та зрозуміти.

Приманки середньої взаємодії є більш просунутими, ніж приманки з низькою взаємодією. Але цього разу від хакера можна отримати більше інформації та більш складні атаки. Оскільки він є більш просунутим, він має більше шляхів входу в безпеку системи, щоб хакер міг отримати доступ до неї. Mwcollect, honeytrap і Nepenthes є одними з приманок середньої взаємодії, які використовуються сьогодні [38].

Honeypots з високою взаємодією є найдосконалішими honeypots. На відміну від приманок із низькою і середньою взаємодією, тут є операційна система. Як наслідок, хакер може зробити що завгодно. Пропорційно більше даних можна отримати від діяльності хакера. Однак він найризикованіший, коли мова йде про безпеку, оскільки надає хакеру такий доступ, що він не має жодних обмежень. Такі приманки забирають багато часу і їх важко обслуговувати. Honeywall є

хорошим прикладом високої взаємодії honeypot.

1.2.3. Перешкоди для розгортання програмних приманок

HPT швидко розвивається і починає застосовуватися у великих організаціях, які зобов’язані захищати дуже конфіденційні або легко замінні товари, такі як гроші. Підприємства та уряди посилюють нагляд за робочим місцем як фізично, так і в цифровому вигляді. Цікаво сформулювати, чи є бар'єри для розгортання технологічними чи організаційними? З технологічної точки зору можна сказати, що більшість розгортань honeypot все ще є дуже експериментальними навіть у комерційно підтримуваних рішеннях [39]. HPT в деяких аспектах, ймовірно, є більш стабільними, ніж пакети системи запобігання вторгненням (IPS), які пропонуються комерційними постачальниками, які використовують деякі сайти. Прийняття IPS може бути наслідком того, що він був проданий як заміна срібної кулі для нездатності IDS виявляти нові форми атак [40]. Технологія IPS, IDS і брандмауер мають зручний і безпечний спосіб відсічі зовнішніх хакерів на периметрі мережевого інтерфейсу.

Навпаки, honeypots є майже антитезою технологій IDS, IPS та брандмауерів, які використовують такий підхід за допомогою принципу, прославленого Сунь-Цзи: тримати своїх ворогів під ретельним наглядом, ніж друзів, вводячи їх усередину оборонного периметру та контролюючи кожну їхню махінацію. Цей режим роботи є суттєвою зміною мислення в порівнянні з ментальністю периметра, на якій базується найсучасніша безпека мережі. Подібним чином сучасна війна переросла в мережеві ефекти і операції, а також безпека мережі потребує аналогічної зміни парадигми від регламентованих відсіків до тактики, орієнтованої на мережу. Однією з найпоширеніших реакцій на розгортання honeypots є юридична концепція «захоплення» [41].

Ця сфера дуже спірна і, звичайно, сильно варіюється в залежності від того, в якій юрисдикції розглядається концепція захоплення. Внутрішні системи honeypot справді створюють різні етичні та моральні проблеми для організації, але не більше, ніж можливі існуючі схеми мережевого моніторингу. Наприклад,

більшість організацій зараз записують електронну пошту користувачів і використання всесвітньої павутини часто без відома користувача. До цих існуючих сценаріїв нагляду на робочому місці можна застосувати аргумент про захоплення. Однак справедливо сказати, що правильно побудовані системи honeypot, розміщені в корпоративній мережі, яка жорстко контролюється відповідною політикою та процедурою, не повинні створювати проблем у цьому відношенні [42]. Honeypot в цій ситуації має стикатися лише з користувачами, які прямо порушують політику компанії та чиї дії є навмисними, зловмисними та навмисними. Це включатиме політику, яка обмежує користувачів на доступ до служб, для яких вони призначені та які законно використовують. Крім того, сама система honeypot міститиме попередження за допомогою банерів або спливаючих повідомлень про те, що доступ до цих систем призначений лише для уповноваженого персоналу і що дії за межами цього пункту порушують політику компанії [43]. Якщо користувач потім вирішує досліджувати або намагається скомпрометувати систему, він прийняв свідоме рішення зробити це, і це не можна назвати захопленням.

На відміну від зовнішніх honeypot, внутрішньо розгорнуті honeypots можуть зафіксувати існуючу поведінку та моделі загроз у межах організаційної мережі. Внутрішнім honeypots, можливо, не доведеться реагувати на нові типи шкідливого коду або нові експлойти, які стосуються зовнішнього honeypot через глибину своєї мережі в організації [44]. Отже, рівень досвіду персоналу, необхідний для ефективного управління одним із них, може бути нижчим, оскільки вони можуть використовувати існуючі дефолти в системах, які пом’якшують відомі загрози. Honeyfiles — це метод, який не пов’язаний з будь-яким конкретним експлойтом і спрямований безпосередньо на визначення того, чи був доступ до файлу.

Внутрішня помилка IDS полягає в тому, що вони шукають певну двійкову послідовність або компрометацію набору правил для окремого екземпляра поведінки, на що вони потім зазвичай реагують однією дією, наприклад розривом з’єднання, яке або вдається, або не вдається [45]. IDS, як правило, мають погані механізми, що дозволяють криміналістичну реконструкцію інциденту поза

межами дій IDS, оскільки їхня основна функція — це захист системи, а не збір даних. Однак HPT призначені для відстеження та моніторингу будь-якої поведінки без шкоди, навіть якщо є взаємодія, наприклад, із IDS у системі honeypot. HPT має бути сконструйовано таким чином, щоб фіксувати дії зловмисників і вплив на систему не тільки на мережевому рівні, а й на рівнях систем і програм. Цей підхід дає змогу фахівцям з безпеки відтворити послідовність подій або дій, які призвели до інциденту, оскільки всі дані повинні бути достатніми для всієї криміналістичної реконструкції інциденту. Це повинно дозволити персоналу внутрішньої безпеки прийняти парадигму навчання щодо усунення інцидентів безпеки шляхом аналізу цих даних.

Внутрішні користувачі також можуть скомпрометувати цінні внутрішні системи, які Інтернет не бачить або не доступний, за допомогою звичайних контрзаходів, не в змозі виявити це зловживання, оскільки вони зазвичай зосереджені на зовнішній точці виходу в Інтернет. Багато організацій використовують системи брандмауера та віртуальної локальної мережі (VLAN) для контролю потоків трафіку до цих високоцінних мереж, а деякі навіть розгортають IDS [46]. Технології IDS і брандмауери можуть страждати від недостатньої повноти судової експертизи через їх спосіб роботи. На організаційному рівні часто діють елементи довіри, які можуть зробити розгортання та навіть керування цими сучасними контрзаходами неприємним [47]. Ця проблема, можливо, посилюється через сприйняття атак як зовнішніх.

Розгортання внутрішньої програмної приманки в цих цінних сегментах мережі може значно зменшити профілі ризиків в організації. Це пояснюється тим, що HPT може виявити поведінку до того, як вона стане проблемою, і працюватиме як система раннього попередження для адміністраторів безпеки [48]. Наприклад, HPT також може ефективно записувати невідомі або невизначені помилки в безпеці існуючих систем, наприклад, неправильні налаштування керування правами для певного сервера або служби, що дозволяє внутрішньому користувачеві отримати доказові можливості.

1.2.4. Внутрішній дизайн програмних приманок

Філософія дизайну внутрішніх приманок у порівнянні зі звичайними зовнішніми приманками впливає на кілька рівнів. По-перше, зовнішні маніпулятори використовують концепцію вичерпання ресурсів зловмисника, що, враховуючи обмежені обмеження широкосмугового Інтернет-з'єднання, є досяжною максимою, хоча вона стає все більш віддаленою зі збільшенням пропускної здатності [49]. Honeypots, які розгорнуті всередині, не мають такої ж можливості, як внутрішній зловмисник, як правило, використовує високошвидкісне з'єднання Ethernet з високим рівнем доступу до honeypot. Затримка мережі зазвичай вимірюється частками мікросекунди і часто в одному сегменті мережі або в безпосередній близькості. Таким чином, проблеми затримки мережі, доступності системи та доступності для збору інформації про атаку або здійснення атаки не є істотним ресурсним бар'єром для внутрішнього зловмисника.

Як згадувалося раніше, внутрішній користувач матиме більш високий рівень початкового аналізу атаки, з якого він може спробувати стежити за системою або навіть проникнути та компроміс [50]. Наприклад, вони можуть використовувати законні інструменти для перегляду мережі, такі як Microsoft Network Neighborhood, щоб перевіряти правила імен серверів, щоб значно зменшити кількість здогадок або зондування, які зовнішній зловмисник повинен зробити, щоб отримати ті самі результати. Це також дозволяє внутрішньому зловмиснику значно звузити вікно виявлення для існуючих контрзаходів, таких як IDS, брандмауери та IPS, в результаті цього зниження активності. Зменшення активності інсайдерів, можливо, не призведе до реакції будь-яких контрзаходів. У деяких випадках контрзаходи будуть налаштовані на довіру до цих внутрішніх об'єктів, що дозволить проводити більш широке та таємне дослідження мережі. Це ще більше зменшується через меншу потребу в фактичній перевірці мережі, щоб виявити такі речі, як операційна система, рівні виправлення, серверні програми та іншу інформацію, яку зовнішній зловмисник збирає перед спробою скомпрометувати систему.

Однією з інших проблем із розгортанням внутрішніх приманок є організаційний витік «секрету» honeypot. Більшість систем honeypot покладаються на обман через маскування, тобто хакер не знає, що вони знаходяться в honeypot. Якщо зловмисний інсайдер тепер знає, що в мережі існує honeypot, він може припинити діяльність або змінити поведінку відповідно до своїх потреб і в кінцевому підсумку не довіряти будь-яким новим додаткам до мережі.

Топологія буде залежати від топології внутрішньої мережі. Зовнішня сторона honeypot може додати шари мережі, такі як демілітаризована зона (DMZ) і VLAN в дизайн [51]. Зазвичай вони призначені для затримки та відволікання зовнішнього хакера; внутрішній дизайн не має такої розкоші. Внутрішній користувач матиме розуміння існуючої топології мережі, і якщо топологія мережі honeypot також не ефективно та надійно імітує існуючу реальну топологію мережі, то знову розкриття внутрішньої мережі honeypot є пропозицією високого ризику. Основну топологію можна побачити на рис. 1.4.

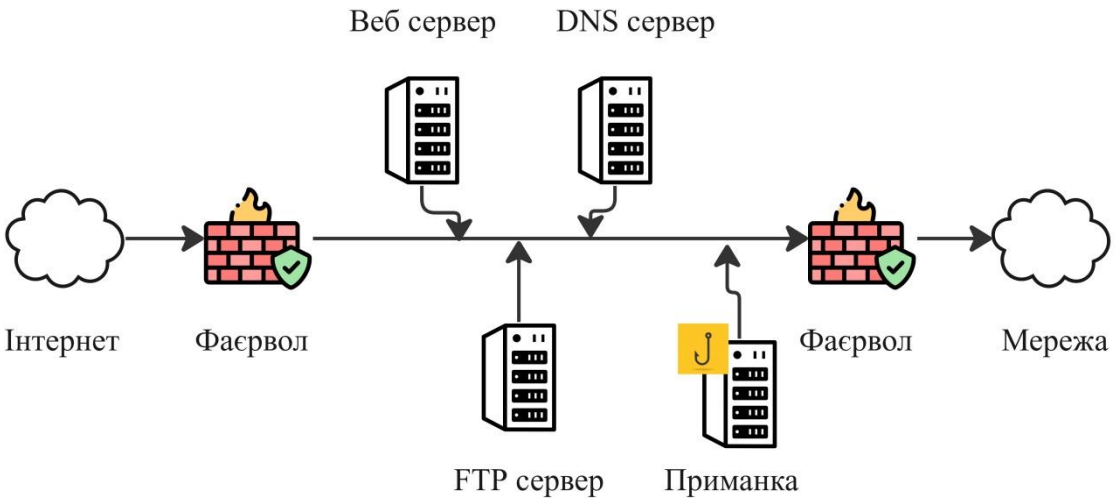

Рис. 1.4. Топологія мережі з програмною приманкою

Ось деякі основні топології, які використовуються для розгортання Honeypot:

- Віртуальна мережа Honeypot (HoneyNet): Це група Honeypot-ів, які розгорнуті у віртуальній мережі та ізольовані від реальної мережі. Це дозволяє зібрати дані про атаки, не ставлячи в ризик реальні системи.

- Внутрішня мережа Honeypot: В цьому сценарії, Honeypot розгортається всередині корпоративної мережі [52]. Це дозволяє виявити внутрішні загрози та атаки, які вже знаходяться всередині мережі.
- Зовнішня мережа Honeypot: Honeypot розгортається поза корпоративною мережею, зазвичай в Демілітаризованій зоні (DMZ) або на хостинг-провайдері. Це допомагає виявити зовнішні атаки та моніторити активність зловмисників в інтернеті.
- Кластер Honeypot: Кілька Honeypot-ів розгортаються разом для створення більш складної мережі з різними рівнями взаємодії. Це може допомогти збирати більше інформації про атаки та надавати більш реалістичні сценарії для зловмисників.
- Розподілена мережа Honeypot: В цьому сценарії, кілька Honeypot-ів розгортаються на різних місцях у мережі, що дозволяє виявити атаки, які цілять на різні частини мережі.

1.3. Проблематика використання приманок та обманок для захисту даних у комп’ютерних мережах

Honeypot можна вважати першим втіленням технології Deception, а з'явилися вони ще в кінці вісімдесятих – початку дев'яностих. Honeypot – це мережевий об’єкт, єдина мета якого – залучити зловмисника та бути атакованим. Коли Honeypot атакують, він реєструє це та зберігає всі дії зловмисника. Надалі ці дані допомагають проаналізувати шлях зловмисника. Друга мета Honeypot — затримати просування зловмисника мережею, змушуючи його витрачати час на вивчення підробленого ресурсу [53]. Представимо схему системи Honeypot на рис. 1.5.

Honeypot може бути повноцінною операційною системою, яка емулює робоче місце співробітника або сервер, або окремий сервіс [54]. Розуміння здібностей зловмисників є важливим для побудови системи захисту, яка може їх виявити.

Рис. 1.5 Схема системи Honeypot

Представимо кілька способів, за допомогою яких зловмисники визначають наявність Honeypots:

- якщо доступ до системи здається занадто простим, можливо, фальшивим;
- зазвичай системи, підключені до Інтернету, не мають непотрібних портів і служб; будь-яке відхилення від цієї конфігурації може свідчити про пастку;
- якщо система все ще має налаштування за замовчуванням, це збільшує ймовірність використання Honeypot;
- якщо на жорсткому диску багато вільного місця або дуже мало програмного забезпечення, можливо, це приманка;
- якщо назви папок тривіальні (наприклад, «Зарплати», «Дані клієнтів», «Паролі»), очевидно, що системи спрямовані на переманювання зловмисників

Усі ці сигнали повідомляють зловмисникам, що система може бути підробкою. Також ці системи мають ряд недоліків:

- потрібно окремо налаштувати кожен фальшивий сервер;
- Honeypots не взаємодіють один з одним і з елементами реальної інфраструктури. Вони не залишають слідів і їх важко виявити хакеру;
- Honeypots, як правило, не об'єднані в централізовану систему.

Ця технологія поступово була замінена іншою, більш досконалою та розумнішою – Deception [55].

Соціальна інженерія та фішингові атаки є прикладом того, як можна обійти будь-який клас рішень, включаючи приманки.

Багато сучасних атак починаються з доставки «приманки» користувачеві,

наприклад фішингового електронного листа, який вони відкривають на своєму комп'ютері. Це дозволяє зловмисному програмному забезпеченню проникнути у внутрішню мережу та дозволяє зловмисникові перейти до планування та виконання наступного етапу напад.

Honeypots не в змозі впоратися з фішинговою атакою так, як це роблять користувачі. Тому Honeypots не зможуть спровокувати та виявити атаку за допомогою такого вектора. На відміну від Honeypots, технології обману наступного покоління можуть автоматично змінювати середовище приманки, не залишаючи його статичним, як і личить реальній мережі, у якій дані користувача та мережі природно змінюються. У той же час технології обману виявляють зловмисника всього за три-чотири кроки в мережі, навіть якщо елементи обману не розгорнуті на кожному вузлі.

Технології обману наступного покоління надають користувачам потужне виявлення атак у режимі реального часу та функціональність криміналістичного збору, практично без помилкових спрацьовувань, і зловмисники ніколи не дізнаються, що вони перебувають під спостереженням. Приманки також ефективні для виявлення зловмисників, але вони мають набагато нижчий рівень виявлення реальних загроз, генерують набагато більше помилкових спрацьовувань і не забезпечують криміналістику з реальних вузлів, які використовують зловмисники для атаки.

Під обманом розуміються рішення класу Intrusion Detection System (IDS) – системи виявлення вторгнень. Основне призначення такої системи – виявлення небажаних спроб доступу до мережі. Іншими словами, Deception допомагає виявляти мережеві атаки. Honeypot – окремий мережевий ресурс , який ні з ким не взаємодіє, а лише чекає, поки зловмисник зафіксує його дії [56]. З іншого боку, Deception технології – це централізована система управління фальшивими мережевими об'єктами, які зазвичай називають пастками (приманками). Кожна пастка, по суті, є окремою приманкою, але всі вони підключені до центрального сервера. Схема Deception технології показана на рис. 1.6.

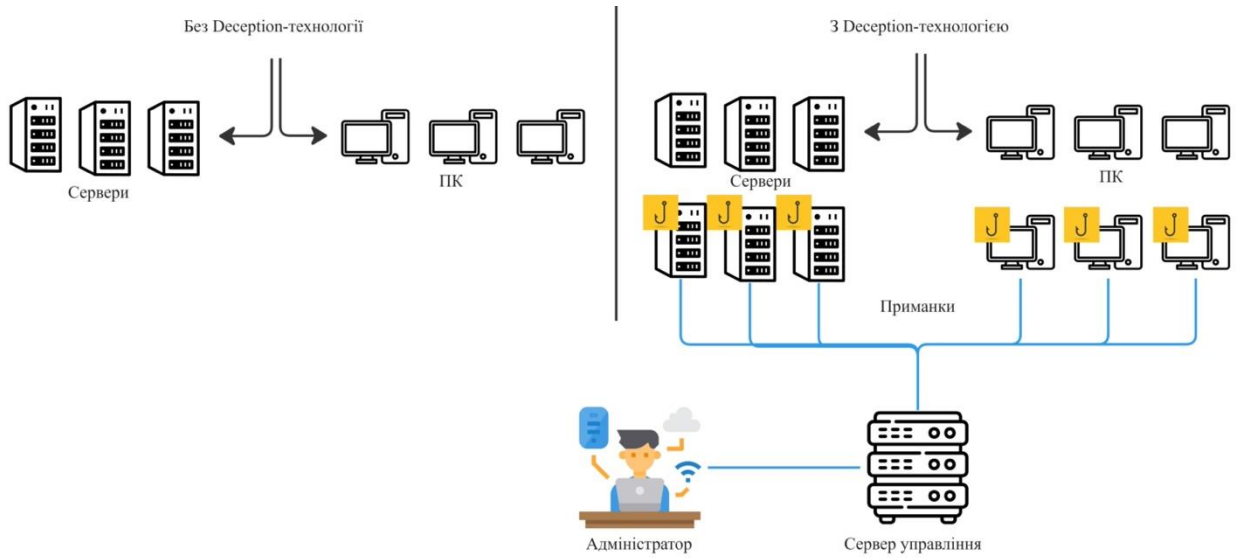

Рис. 1.6. Схема системи обману

Такі рішення зазвичай мають зручний інтерфейс для управління пастками. Оператор може створити пастки з потрібним набором емульованих мережевих служб, у вибраній підмережі, з потрібним способом отримання IP-адреси тощо [57]. Пастки та емульовані на них служби підтримують постійний зв’язок із сервером. Як і Honeypots, пастки в Deception не забезпечують законної взаємодії з мережею (за винятком взаємодії з іншими компонентами Deception). Пастка повідомлятиме серверу про будь-які спроби взаємодії з нею: це служить індикатором атаки. У цьому випадку оператор може миттєво отримати повідомлення про подію [58]. У ньому будуть вказані подробиці того, що сталося: адреса і порт джерела і цілі, протокол взаємодії, час відповіді і так далі. Додаткові модулі в Deception технології також можуть надавати ручні або автоматизовані можливості реагування на інциденти (рис. 1.7).

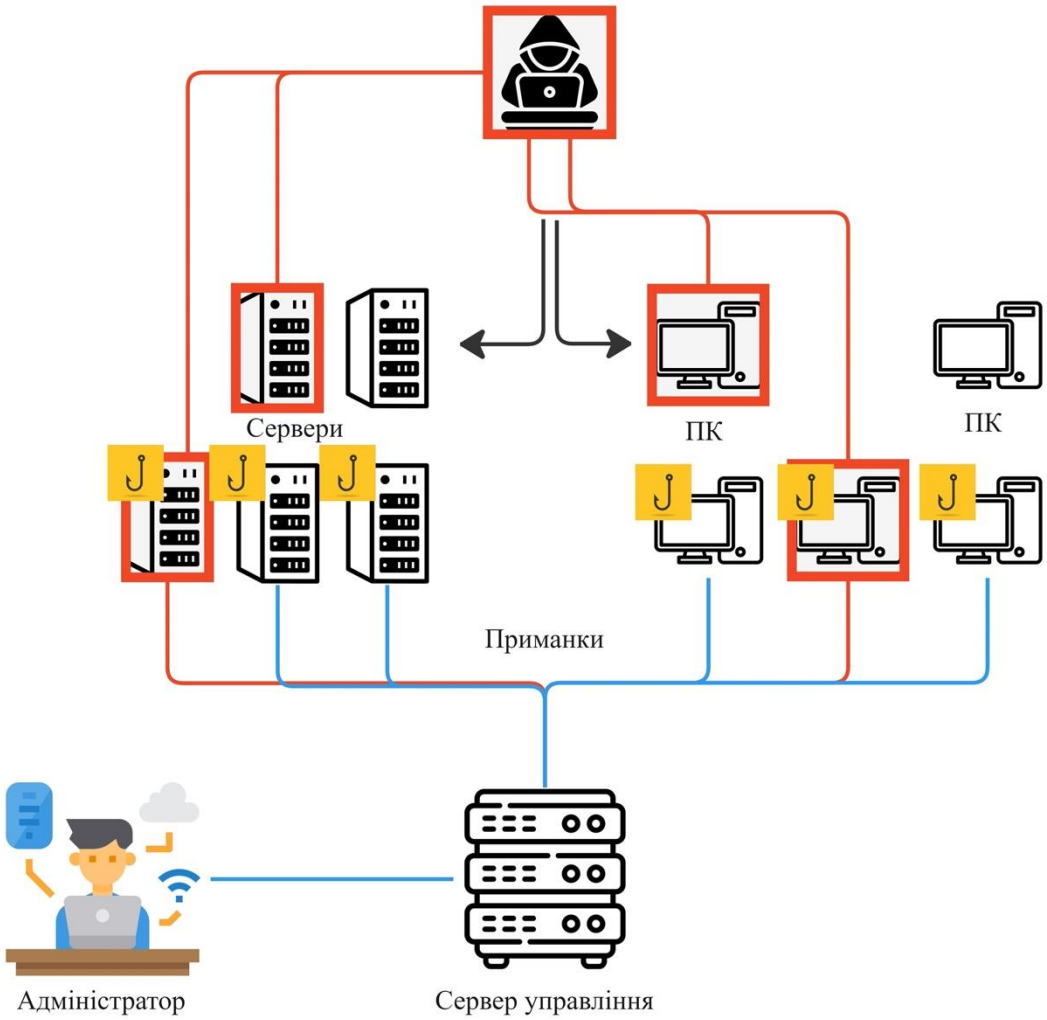

Рис. 1.7. Схема системи приманки

Поняття обману може включати й інші речі. Деякі компоненти допомагають спростити конфігурацію та автоматизацію розгортання, інші роблять пастки більш схожими на справжні мережеві служби, а треті привертають увагу хакерів до фальшивих цілей [59]. Деякі компоненти можуть виконувати пов’язані завдання, наприклад реагувати на інциденти, збирати індикатори компрометації з робочих станцій і шукати на них уразливе програмне забезпечення.

Агент – це програма, яка встановлюється на реальних робочих станціях або серверах користувачів. Вона здатна спілкуватися з сервером обману, виконувати його команди або передавати дані користувача в центр управління. Серед рішень класу Deception є як продукти, що містять агента, так і ті, що обходяться без нього (рис. 1.8).

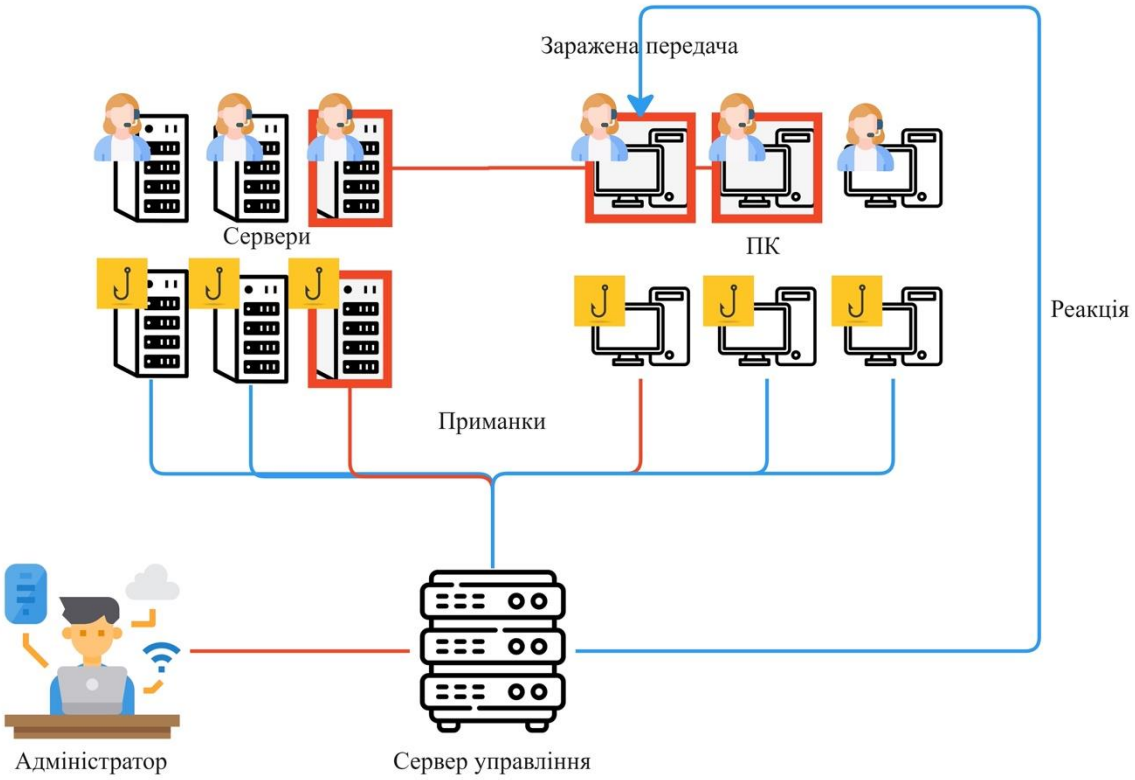

Рис. 1.8. Схема агентської системи

Завдання агентам можуть включати:

- збір даних про стан АРМ;
- роздача приманок;
- емуляція активності в мережі;
- реагування на інцидент (ручне або автоматизоване);
- збір даних для криміналістики;
- інше - в міру потреб замовників і фантазії забудовника.

Діяльність агентів повинна бути прихованою від людини, яка працює за комп'ютером [60]. По-перше, користувач може навмисно або випадково видалити агент або його компоненти. По-друге, наявність на робочій станції невідомого (або певною мірою відомого - якщо користувач про це попереджений) програмного забезпечення може викликати відчуття дискомфорту. По-третє, все, що бачить користувач, побачить зловмисник, який отримав доступ до цього комп'ютера.

Агентські рішення в рамках обману повинні прийматися таким чином, щоб

користувач не бачив ні агента, ні слідів його життєдіяльності (або хоча б намагався мінімізувати це). Тому агенти зазвичай працюють у привілейованому режимі, як драйвер для Windows або модуль ядра для Linux. Це дозволяє, наприклад, перехоплювати системні виклики, щоб забезпечити прихованість, а також не дозволяє користувачеві видалити агента або запобігти його роботі [61].

Honeypot містить посилання та дані для доступу до підробленого мережевого ресурсу. Зловмисник, знайшовши таке посилання і дані авторизації, звичайно, хоче перевірити, що це за сервіс. Він потрапляє в пастку, а потім спрацьовує сигнал про подію (рис. 1.9).

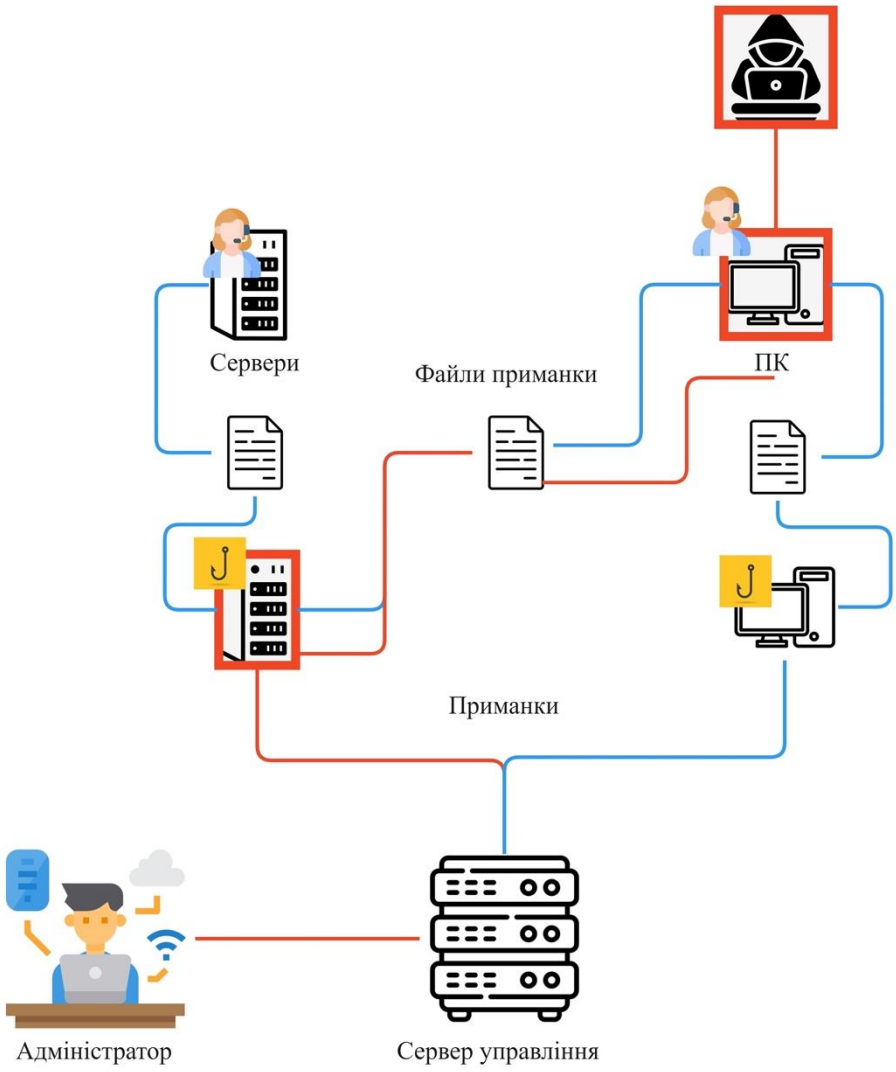

Рис. 1.9. Приманка, агент і наживка в системі

Види і способи розміщення приманки залежать від типу пастки, до якої веде приманка. Приманки можна поширювати декількома способами. Якщо в обмані присутні агенти, на них покладається завдання розкидати приманки [62]. У цьому випадку процес можна легко автоматизувати: керуючий сервер надсилає команду агенту, а останній виконує необхідні дії для встановлення приманки.

Для деяких приманок може знадобитися поштова адреса, адреса домену або щось інше. Проблему можна вирішити шляхом ведення бази даних фальшивих користувачів мережі. Існують різні підходи до ведення такої бази даних (рис. 1.10).

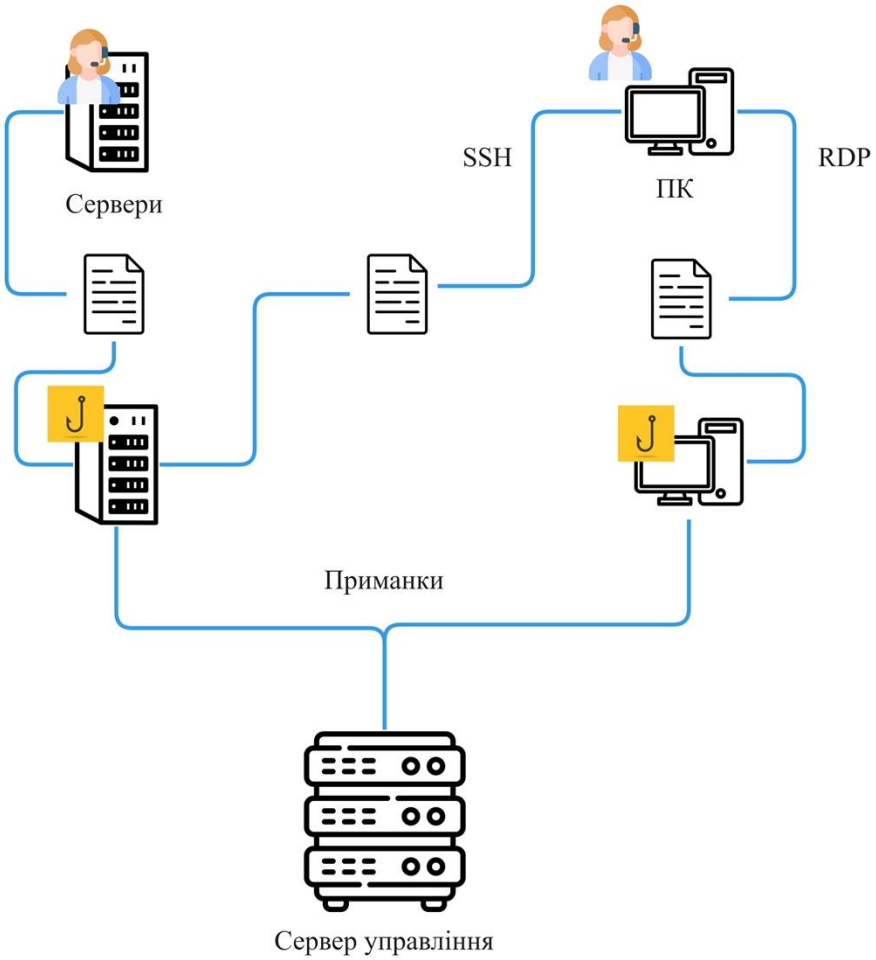

Рис. 1.10. Схема системи помилкових користувачів

Наприклад, Deception можна інтегрувати з системою аналізу трафіку. Це дає змогу розпізнавати наявність авторизаційних даних у мережевому трафіку,

знаходити в них спільні риси та генерувати користувачів, схожих на справжніх, за ідентифікованими правилами [63].

NFC-технологія може бути використана разом з програмними приманками або технологіями обману, такими як Honeypots, для підвищення рівня безпеки комп'ютерних мереж, особливо в контексті автентифікації та авторизації осіб, наприклад:

1. Підвищення рівня безпеки автентифікації: Використання NFC-міток у якості складової подвійної автентифікації (пароль + мітка) забезпечує додатковий рівень захисту для децентралізованих хостів та комп'ютерних мереж, що зменшує ймовірність несанкціонованого доступу до системи.

2. Захист від фішингу та інших атак: Використання NFC-міток в поєднанні з паролями може захистити користувачів від фішингу, брутфорс-атак та інших методів, що спрямовані на крадіжку паролів. Навіть якщо зловмисник отримає доступ до пароля, без фізичного доступу до NFC-мітки він не зможе увійти до системи.

3. Відволікання зловмисників: Використання NFC-міток в програмних приманках та Honeypots може спонукати зловмисників витрачати свій час та ресурси на аналіз та атаку приманок, замість того, щоб намагатися отримати доступ до реальних систем. Це допоможе забезпечити безпеку справжніх активів корпорації.

4. Збір інформації про зловмисників: За допомогою NFC-технологій можна створити приманку, яка зберігає інформацію про зловмисників та їх дії. Це дозволить збирати даніпро активність зловмисників, забезпечуючи знання про загрози та можливість здійснювати превентивні заходи для захисту комп'ютерних мереж. Аналіз отриманих даних може допомогти виявити слабкі місця в системі та забезпечити своєчасне їх усунення.

5. Безконтактне спілкування: NFC-технологія працює на основі безконтактного спілкування між пристроями на коротких відстанях, що забезпечує швидкість та зручність використання. Це може бути використано для створення приманок, які легко розгортаються та контролюються, одночасно

забезпечуючи надійний захист від несанкціонованого доступу.

6. Захист від віддаленого перехоплення: Оскільки NFC має коротку діапазон дії, зазвичай менше 10 сантиметрів, це ускладнює можливість віддаленого перехоплення сигналу зловмисниками. Це означає, що зловмисникам потрібно мати фізичний доступ до приманки або мітки для виконання атаки, що підвищує рівень безпеки системи.

Узагальнюючи, використання NFC-технології разом з програмними приманками та технологіями обману, такими як Honeypots, може підвищити рівень безпеки комп'ютерних мереж і децентралізованих хостів. Подвійна автентифікація з паролем та NFC-міткою, а також відволікання зловмисників, збір інформації про них та надійний захист від віддалених атак роблять NFC-технологію цінним інструментом для підвищення безпеки інформаційних систем.

1.4. Характеристика Blockchain технології

Blockchain — це розподілені мережі, які можуть мати мільйони користувачів у всьому світі. Кожен користувач може додавати інформацію в Blockchain, і всі дані в Blockchain захищені за допомогою криптографії. Кожен інший учасник мережі відповідає за перевірку того, що дані, які додаються в Blockchain, є реальними [64]. Це робиться за допомогою системи з трьох ключів (приватний, відкритий і ключ одержувача), які дозволяють учасникам перевіряти правдивість даних, а також підтверджувати, від кого вони походять, як зображено на рис. 1.11.

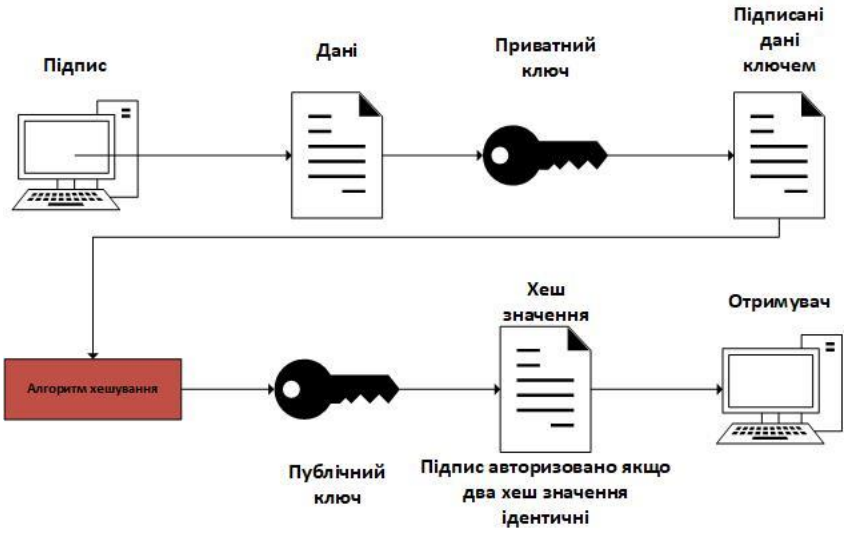

Рис. 1.11. Шифрування передачі даних в Blockchain

Щоб оновити певний фрагмент даних, власник цих даних повинен додати новий блок поверх попереднього блоку, створюючи дуже специфічний ланцюжок коду. Якщо щось, навіть таке невелике, як кома, змінюється від того, як воно відображається в попередньому блоці, весь ланцюжок у мережі також змінюється відповідно [65]. Це означає, що кожна окрема зміна або зміна будь-якої частини даних відстежується, і абсолютно жодні дані не втрачаються чи видаляються, оскільки користувачі завжди можуть переглянути попередні версії блоку, щоб визначити, що відрізняється в останній версії. Використання цієї ретельної форми ведення записів дозволяє системі легко виявляти блоки, які містять неправильні або хибні дані, запобігаючи втраті, пошкодженню та пошкодженню, як зображено на рис. 1. 12.

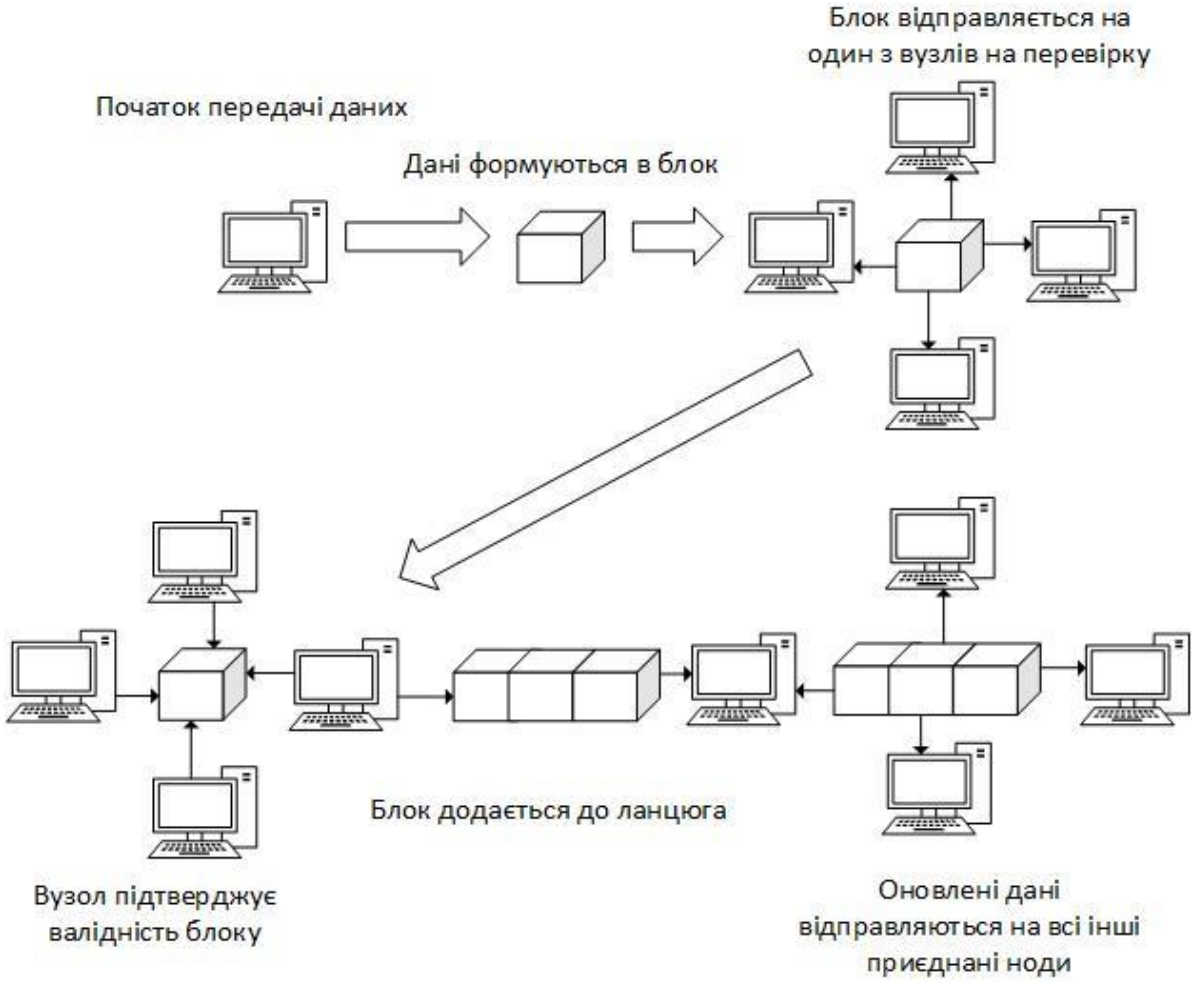

Рис. 1.12. Схема перевірки блоку на вузлах

Ще одна важлива річ, яку слід зазначити щодо користувачів Blockchain, це те, що вони можуть зберігати всі дані в своїй мережі на своєму комп'ютері, якщо захочуть (а дуже багато з них це роблять). Це призводить до двох речей. По-перше, вони можуть заробити гроші на оренді свого «зайвого» складського приміщення, а по-друге, вони гарантують, що ланцюжок не зруйнується. Якщо, наприклад, хтось, хто не є власником частини даних (скажімо, хакер), намагається втрутитися в блок, вся система аналізує кожен блок даних, щоб знайти той, який відрізняється від решти (або від більшість). Якщо система знаходить цей тип блоку, вона просто виключає його з ланцюга, ідентифікуючи його як хибний [66].

Технологія Blockchain розроблена таким чином, що немає центрального органу чи місця зберігання. Кожен користувач мережі відіграє певну роль у збереженні частини або всього Blockchain. Кожен несе відповідальність за перевірку даних, які зберігаються та/або надаються, щоб переконатися, що неправдиві дані не можуть бути додані, а наявні дані неможливо видалити, як зображено на рис. 1.13.

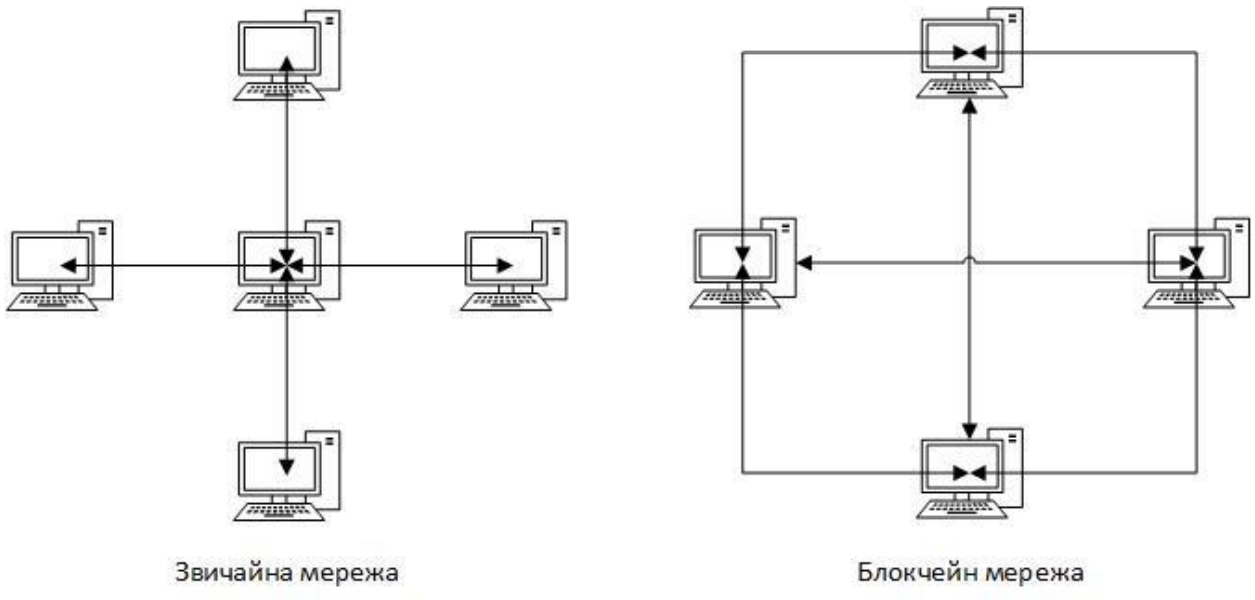

Рис. 1.13. Порівняння звичайної централізованої мережі з мережею Blockchain

Технологія Blockchain є одним із найкращих інструментів, які ми зараз маємо для захисту даних від хакерів, запобігання потенційному шахрайству та зниження ймовірності викрадення або скомпрометації даних.

Щоб знищити або зіпсувати Blockchain, хакеру доведеться знищити дані, що зберігаються на комп'ютері кожного користувача в глобальній мережі. Це можуть бути мільйони комп'ютерів, кожен з яких зберігає копію деяких або всіх даних. Якщо хакер не зміг одночасно зруйнувати всю мережу (що майже неможливо), непошкоджені комп'ютери, також відомі як «вузли», продовжуватимуть працювати, щоб перевірити та вести запис усіх даних у мережі. Неможливість виконання завдання, наприклад знищення цілого ланцюжка, збільшується разом із кількістю користувачів у мережі [67]. Більші мережі Blockchain з більшою кількістю користувачів мають нескінченно нижчий ризик зазнати атаки хакерів через складність, необхідну для проникнення в таку мережу.

1.4.1. Технологія Blockchain

Blockchain — це однорангова (p2p) розподілена база даних, яка підтримує список постійно зростаючих записів, які називаються блоками, вони між собою пов'язані та захищені, як правило, за допомогою криптографії з відкритим ключем. Завдяки технології Blockchain нова інформація додається до блоку та стає доступною для всіх вузлів у розподіленій мережі замість додавання до централізованої бази даних у традиційній централізованій системі [68]. Кожен блок у Blockchain ідентифікується хеш-значенням, яке зазвичай генерується за допомогою безпечного хеш-криптографічного алгоритму 256 біт (SHA256). Хеш-значення поточного заголовка блоку (батьківського) зв'язується та зберігається в наступному блоці (дочірньому), як показано на рис. 1.14; отже, якщо буде зміна вмісту будь-якого блоку, його хеш також буде відповідно змінено, і ця зміна буде поширена по всій мережі, щоб зробити цей блок недійсним.

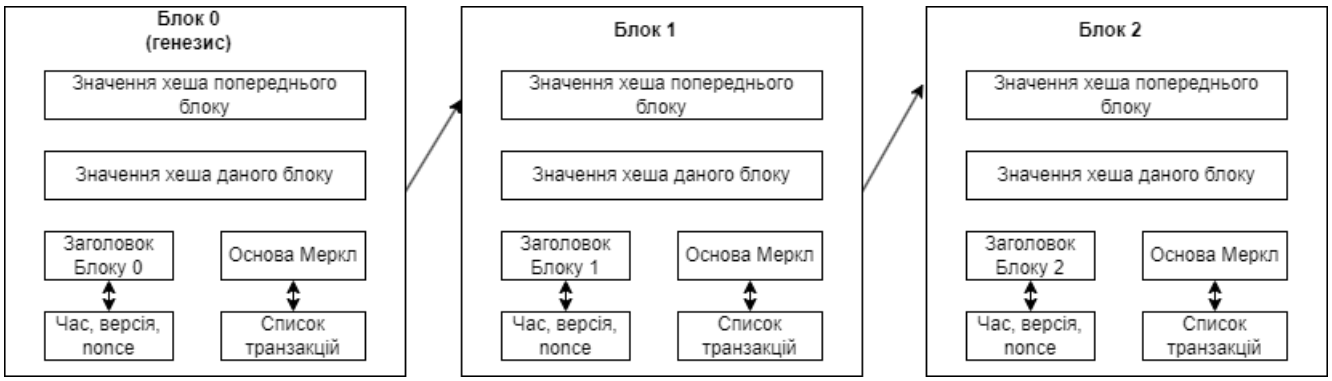

Рис. 1.14. Приклад блоків Blockchain та їх зв’язок

Завдяки цьому механізму технологія Blockchain не потребує посередника чи довіреної третьої сторони, оскільки вона є децентралізованою та розподіленою. Учасники Blockchain мають закриті ключі, призначені їм для цифрового підпису та підтвердження транзакцій, які вони здійснюють.

Як показано на рис. 12, блок складається із заголовка, що містить метадані, і довгого списку транзакцій, виконаних у цьому блоці. Заголовок блоку зазвичай містить мітку часу, одноразовий номер, версію та доказ складності. Мітка часу вказує на час створення блоку; nonce — це випадкове число, згенероване консенсусним алгоритмом для обчислення хеш-значення блоку; версія вказує на номер версії Blockchain, а доказом складності є згенероване хеш-значення, яке має бути меншим за поточне цільове хеш-значення.

Перший блок, відомий як блок генезису, жорстко закодований шляхом вбудовування деяких випадкових даних у програму Blockchain [69]. Незважаючи на те, що кожен блок має лише одного батьківського й одного дочірнього, дійсний блок може мати двох або більше дочірніх елементів, тимчасово створених, коли до блоку одночасно додаються два або більше вузлів (мережевих однорангових вузлів), що призводить до двох або кількох гілок від одного батьківського блоку. Цю ситуацію зазвичай називають «розгалуженням» і усувають, приймаючи ланцюжок, який стає довшим за інші, як дійсний Blockchain, і роблячи всі інші, коротші, недійсними, із ситуацією з двома розгалуженнями, продемонстрованою на рис. 1.15. Сформовані гілки мають однакову довжину; у цій ситуації процес

додавання нових блоків продовжується для всіх ланцюжків, які підлягають перевірці, доки одна гілка не стане довшою за інші, таким чином, дійсними.

У блоці всі транзакції пов’язані між собою за допомогою дерева Меркла. Дерево Merkle — це перевернуте бінарне дерево, яке використовується технологією Blockchain для узагальнення всіх транзакцій у блоці.

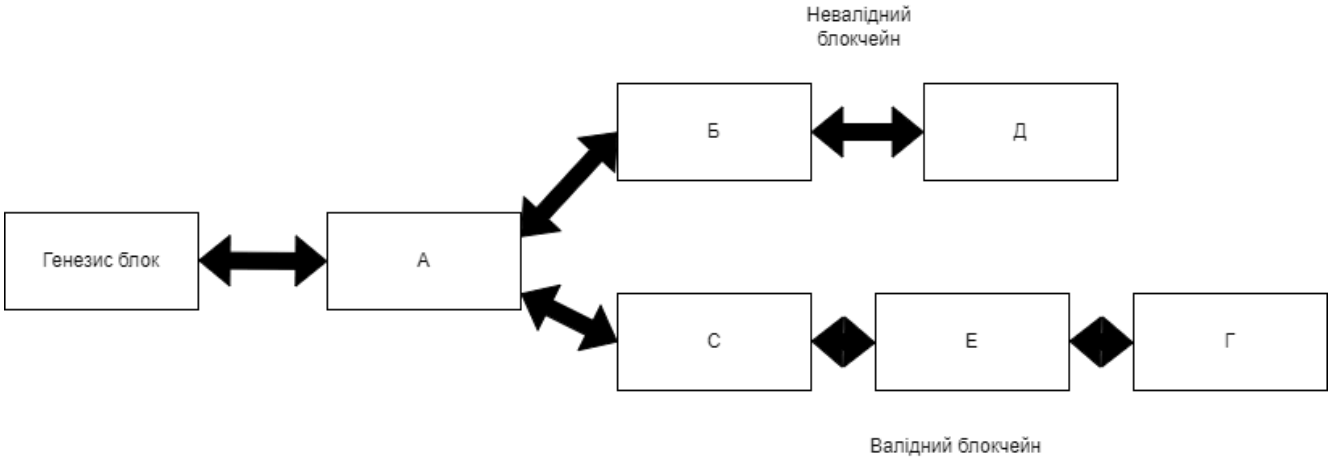

Рис. 1.15. Перевірка Blockchain для розгалуження

Щоб побудувати дерево Меркла, пару транзакцій рекурсивно хешують, поки вони не сформують лише один кореневий вузол у верхній частині дерева, який називається коренем Меркла , як показано на рис. 1.16. Точніше, корінь Меркла — це хеш усіх транзакцій, які складають блок у мережі Blockchain. Будь-яка незначна зміна даних змінить кореневий хеш Меркла, що призведе до недійсного запису. Найпоширенішим криптографічним хеш-алгоритмом, який використовується для побудови дерева Меркла, є безпечний 256-бітовий хеш-алгоритм (SHA256) [70]. Якщо є непарна кількість транзакцій, хеш останньої транзакції дублюється, щоб створити парну кількість транзакцій, таким чином утворюючи збалансоване дерево.

Вузли в мережі Blockchain запускають консенсусний алгоритм для перевірки транзакцій. Існує кілька консенсусних алгоритмів (протоколів), доступних для технології Blockchain, таких як доказ роботи (PoW), доказ частки (PoS), делегований доказ частки (DPoS) і доказ складності (PoD). Наприклад, Bitcoin використовує PoW, тоді як Ethereum і Bitshare реалізують PoS і DPoS

відповідно. У PoW вузли-майнери, які хочуть додати (видобути) новий блок до мережі Blockchain, повинні спочатку вирішити складну математичну головоломку, яка потребує великої обчислювальної потужності.

Blockchain-мережі можуть бути дозволеними або без дозволів. Мережа без дозволу або загальнодоступний Blockchain дозволяє будь-якому користувачеві створити особисту адресу, приєднатися до мережі та брати участь у консенсусі, тоді як дозволена або приватна мережа дозволяє приєднатися лише кільком обмеженим вузлам.

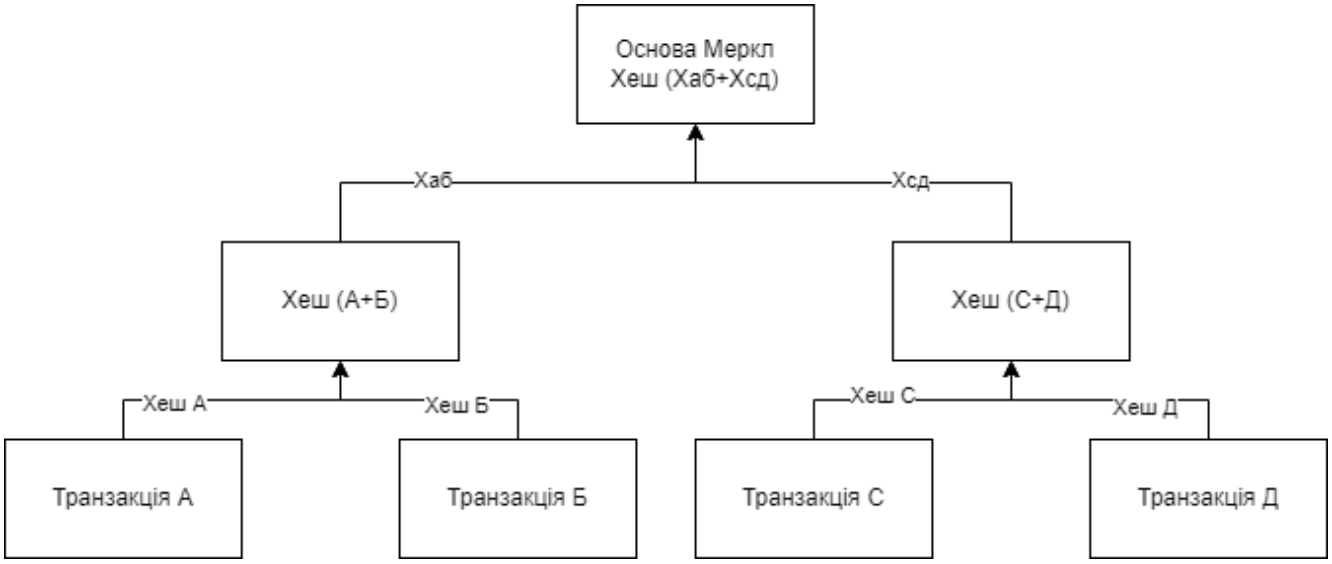

Рис. 1.16. Ілюстрація дерева Меркла

1.5. Аналіз використання Blockchain технології

Blockchain є базовою технологією, на якій працюють багато криптовалюти, такі як біткойн та Ethereum, але його унікальний спосіб безпечного запису та передачі інформації має більш широке застосування за межами криптовалюти.

Blockchain — це тип розподілених ланцюгів. Технологія розподілених ланцюгів (DLT) дозволяє вести записи на кількох комп’ютерах, відомих як «вузли». Будь-який користувач Blockchain може бути вузлом, але для роботи потрібно багато потужності комп’ютера. Вузли перевіряють, затверджують і зберігають дані в реєстрі. Це відрізняється від традиційних методів ведення записів, які зберігають дані в центральному місці, наприклад, на комп’ютерному сервері.

Blockchain організовує інформацію, додану до книги, у блоки або групи даних. Кожен блок може містити лише певну кількість інформації, тому нові блоки постійно додаються до книги, утворюючи ланцюжок.

Кожен блок має свій унікальний ідентифікатор, криптографічний «хеш». Хеш не тільки захищає інформацію в блоці від будь-кого без необхідного коду, але також захищає місце блоку вздовж ланцюжка, ідентифікуючи блок, який був перед ним.

Криптографічний хеш — це «набір цифр і букв, довжина яких може складати до 64 цифр», — каже Вікас Агарвал, партнер консультативної практики PwC у сфері фінансових послуг. «Це унікальний код, який дозволяє шматочкам головоломки поєднуватися між собою».

Після того, як інформація додається в Blockchain і зашифрована за допомогою хеша, вона стає постійною і незмінною. Кожен вузол має власний запис повної часової шкали даних уздовж Blockchain, повертаючись до її початку. Якщо хтось підробив або зламав один комп’ютер і маніпулював даними для власної вигоди, це не змінить інформацію, що зберігається на інших вузлах. Змінений запис можна легко розрізнити та виправити, оскільки він не відповідає більшості.

Те, як працює система, практично неможливо для когось відтворити обчислювальну потужність, яка відбувається на задній панелі, щоб здійснити її зворотний інжиніринг і якимось чином з’ясувати, що це за хеші.

Ось приклад того, як Blockchain використовується для перевірки та запису транзакцій біткойн.

1. Споживач купує біткойн.
2. Дані транзакції надсилаються через децентралізовану мережу вузлів Bitcoin.
3. Вузли підтверджують транзакцію.
4. Після схвалення транзакція групується з іншими транзакціями, утворюючи блок, який додається до постійно зростаючого ланцюга транзакцій.
5. Завершений блок зашифрований, а запис транзакції постійний; його не можна видалити чи змінити на Blockchain.

Blockchain біткоїна є загальнодоступним, що означає, що кожен, хто володіє біткойн, може переглядати запис транзакції. Хоча відстежити особу облікового запису може бути важко, запис показує, які облікові записи здійснюють транзакції в Blockchain. Публічні Blockchain також дозволяють будь-якому користувачеві з необхідною потужністю комп'ютера брати участь у затвердженні та запису транзакцій у Blockchain як вузол.

Але не всі Blockchain є загальнодоступними. Blockchain можуть бути розроблені як приватні системи, тому власник може обмежити, хто може вносити зміни або доповнення до Blockchain. Хоча пул учасників може бути меншим у приватному Blockchain, він все ще децентралізований серед тих, хто бере участь. Приватні Blockchain забезпечують безпеку будь-яких даних, що зберігаються в базі даних, використовуючи ті самі методи шифрування.

Ідея безпечного, децентралізованого постійного запису інформації викликала інтерес у ряді галузей і потенційно містить рішення для багатьох проблем безпеки, процесів ведення записів і проблем володіння даними, з якими ми стикаємося сьогодні. Blockchain дає нам технологію безпечного переміщення інформації, каже Агарвал, і мати майже повну впевненість у достовірності будь-якої інформації, яку ви хочете захистити. Розглянемо, наприклад, історії, які поширювалися останніми тижнями про мемів і знаменитостей, які заробляли на цифровій власності, продаючи NFT (незамінні токени). Оскільки основний запис Blockchain є незмінним, NFT дозволяють продавцям підтвердити справжність цифрового активу. Коли ви купуєте NFT, ця транзакція додається до книги Blockchain та стає підтвердженим записом власності. Для тих, хто хоче мати можливість перевірити справжність цифрової роботи, Blockchain допомагає цінувати цифрове мистецтво та предмети колекціонування так само, як і їхні фізичні аналоги. Теоретично це призводить до того, що творці зберігають цінність за допомогою речей, які отримують гонорар за копії цифрового мистецтва.

Подібне використання ілюструє привабливість Blockchain не тільки для безпеки, але й цілісністю інформації. Blockchain має потенціал, щоб дати людям більше безпеки та впевненості в цьому. Компанії та уряди по всьому світу

продовжують тестувати та впроваджувати технологію Blockchain, але це не станеться миттєво. Якщо ми коли-небудь досягнемо точки, коли державна валюта буде базуватися на Blockchain або медичні записи будуть конвертовані в Blockchain, це не скоро.

1.5.1. Використання Blockchain систем як захисту на реальних прикладах

Біткойн також працює за технологією Blockchain. Весь Blockchain зберігається у величезній мережі комп'ютерів. Отже, жодна людина не має контролю над історією. Скажімо, користувач дізнається дані, пов'язані з криптовалютою, від Crypto Head [71], купує біткойни та оплачує іншим, використовуючи біткойни. Що буде далі? Комп'ютери на Blockchain Bitcoin поспішають перевірити точність транзакції. Цей крок засвідчує все, що сталося в ланцюжку. Таким чином, ніхто не може повернутися назад і змінити речі. Тому Blockchain не можна легко підробити.

Ця складна структура забезпечує технологію Blockchain з можливістю бути найбезпечнішою формою зберігання та обміну інформацією в Інтернеті, яку ми виявили. Саме тому новатори почали застосовувати технологію в різних секторах, щоб запобігти шахрайству та підвищити захист даних.

Компанія Guardtime знімає необхідність використання ключів для перевірки. Замість цього вони розподіляють кожну частину даних між вузлами всієї системи. Якщо хтось намагається змінити дані, система аналізує всю масу ланцюжків, порівнює їх з пакетом метаданих, а потім виключає всі, що не збігаються.

Це означає, що єдиний спосіб стерти весь Blockchain — це знищити кожен окремий вузол. Якщо лише один вузол продовжує працювати з правильними даними, всю систему можна відновити, навіть якщо всі інші вузли скомпрометовані. В умовах гібридних воєн це дозволяє убезпечити будь-яку конфіденційну інформацію від знищення або спотворення.

Система Guardtime працює таким чином, що вона завжди може визначити, коли в дані були внесені зміни, і постійно перевіряє зміни. Це гарантує, що немає дискретного способу втрутитися в блоки в ланцюжку, а дані залишаються безкомпромісними.

При виборі готових рішень або при створенні власного першорівневого Blockchain необхідно розуміти як вони формуються та з чого складають. В таблиці 1.2 наведено порівняння декількох найпопулярніших першорівневих Blockchain систем.

Таблиця 1.2.

Порівняльна характеристика додатків побудованих на першому рівні Blockchain

Характеристики	Bitcoin	Etherum	IOTA
Повністю Випущена	Так	Так	Так
Дозволи майнити	Публічний	Публічний, приватний, гібрид	Публічний
Система безпечних операцій	Так	Так	Є довірені ноди-валідатори
Підтримка зовнішніх додатків	Лише фінансові	Так	Так
Конфіденційність даних	Ні	Ні	Так
Можливість розгалуження	Так	Так	Ні
Відсутність комісії	Ні	Ні	Так
Керування ключами	Ні	Ні	Ні
Керування персональними ідентифікаторами	Ні	Ні	Ні

Автентифікація користувача	Цифровий підпис	Цифровий підпис	Цифровий підпис

Висновки до 1 розділу

Таким чином, у першому розділі дисертаційної роботи було проведено аналіз наукової літератури за темою дисертації, зокрема проаналізовано сучасні підходи до побудови систем з використанням програмних приманок та проаналізовано системи Blockchain. Проведене дослідження дозволяє виділити наступне:

1. Проведений аналіз використання приманок як ефективної протидії зовнішньому нападу показав, що їхнє використання є добре перевіреною концепцією. Велика частина літератури в цій галузі досліджень зосереджується на використанні приманок в якості зовнішньої протидії зовнішнім атакам. Існує дуже мало доступної літератури або експериментів, які проводяться з внутрішньо орієнтованими приманками. Це свідчить про необхідність певних пошукових і постійних досліджень у цій галузі.

2. Дослідження способу роботи внутрішнього розгортання програмних приманок представляє значні зміни у фокусі та дизайні зовнішніх програмних приманок. Шкідливі інсайдери матимуть значну тактичну перевагу над своїми зовнішніми колегами під час зондування, проникнення або компрометації мережі, що підтверджується багатьма опитуваннями безпеки, які проводяться в усьому світі. Це суттєво змінює деякі основні передумови, на яких розгортаються існуючі технології прогрманих приманок, і має значний вплив на проектування та розгортання. Також програмні приманки не являються основним захисним механізмом, а слугують лише для того, щоб збирати дані про атаки зловмисників. Підсилення цієї технології в аспектах захисту, а не лише моніторингу, дозволить створити новий потужний безпековий шар для комп’ютерної мережі та покращити її захисні спроможності.

3. Аналіз внутрішніх програмних приманок показав, що вони пропонують потенційно життєздатний метод відстеження шкідливих інсайдерів у порівнянні з іншими сучасними мережевими контрзаходами, і це вимагає подальшого

розслідування. Високий рівень взаємодії та ведення журналів, який можливий в одному екземплярі, потенційно перевершує брандмауери, IDS та IPS. Там, де атаки можуть бути просто відхилені брандмауерами або IDS, система приманок дозволить посилити спостереження та моніторинг діяльності зловмисних інсайдерів. Майбутні дослідження в цій галузі мають бути проведені щодо практичної реалізації та розгортання внутрішніх маніпуляцій у сучасних організаційних умовах. Розгортання внутрішніх приманок є непростою технічною проблемою, але також має потенційно багато організаційних факторів, таких як довіра та інсайдерські зловживання, з якими можна боротися позбавляючи права на цю конкретну технологію.

4. Дослідження показало, що інноваційні види використання технології Blockchain вже стають частиною інших галузей, крім криптовалют, наприклад в навчальних сферах та урядових сферах. Використання Blockchain технологій дозволяє покращити системи електронного урядування, що збільшить зручність користування даною системою, швидкість передавання даних та підвищить загальний безпековий рівень системи.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ВИКОРИСТАННЯ ВЛАСТИВОСТЕЙ BLOCKCHAIN ДЛЯ РОЗРОБЛЕННЯ КОНЦЕПТУАЛЬНОГО ПІДХОДУ З ВИКОРИСТАННЯ ТЕХНОЛОГІЇ ДЕЦЕНТРАЛІЗАЦІЇ В РАМКАХ КІБЕРЗАХИСТУ

2.1. Дослідження можливостей використання Blockchain для захисту даних на об’єктах інфраструктури

Blockchain перетворився на одну з найбільш надійних форм транзакцій у сфері цифрових мереж. Як розроблено та задумано, технологія була визнана за забезпечення цілісності інформації. Якщо його правильно використовувати, багато секторів можуть отримати від нього вигоду, як зображено на рис. 2.1.

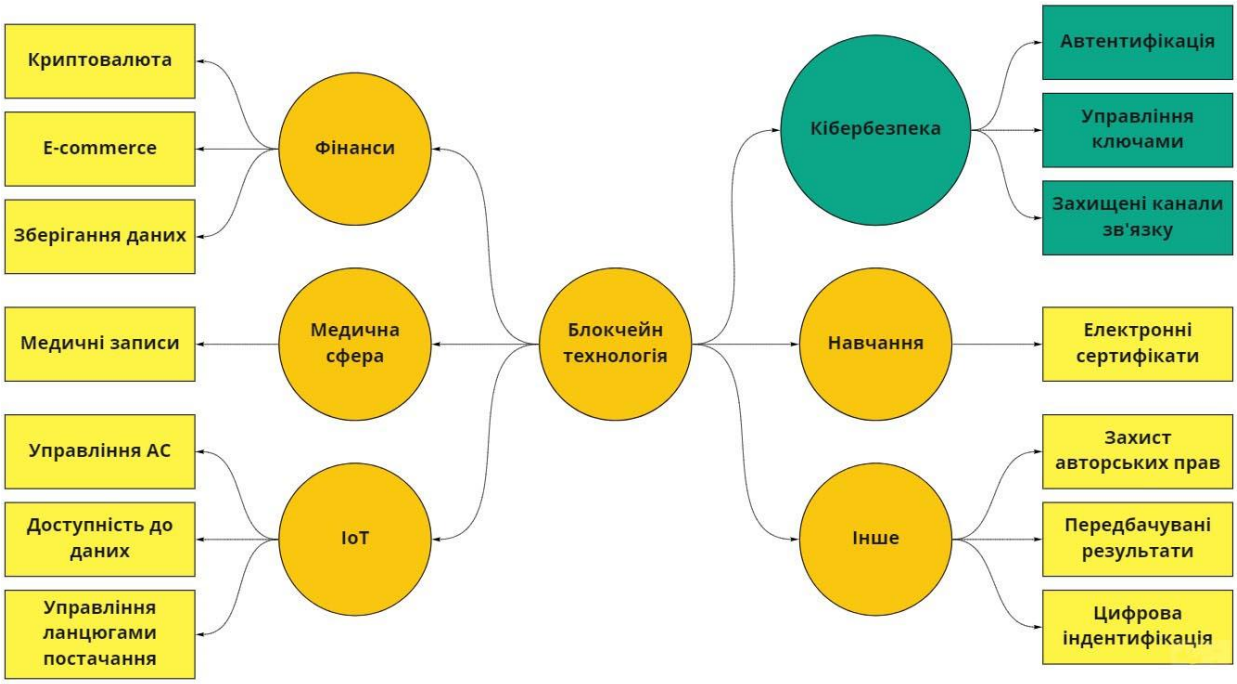

Рис. 2.1. Сектори застосування Blockchain технології

Маючи потенціал бути практичним для багатьох застосувань, Blockchain може бути реалізований у багатьох випадках. Одним з найкращих застосувань було б використання його гарантії цілісності для створення рішень кібербезпеки для багатьох інших технологій. Нижче наведено кілька випадків використання майбутнього вигідного використання Blockchain для посилення кібербезпеки:

- Захист приватних повідомлень. Оскільки Інтернет перетворює світ на глобальне місто, все більше людей приєднуються до соціальних мереж. Кількість соціальних мереж також зростає. З кожним світанком запускається все більше соціальних додатків, оскільки розмовна комерція набирає популярності. Під час цих взаємодій збирається величезна кількість метаданих. Більшість користувачів платформ соціальних мереж захищають служби та свої дані за допомогою слабких, ненадійних паролів.

Більшість компаній із обміну повідомленнями звертаються до Blockchain для захисту даних користувачів як кращого варіанту, ніж наскрізне шифрування, яке вони зараз використовують. Blockchain можна використовувати для створення стандартного протоколу безпеки. Для забезпечення можливостей міжмесенджерного зв'язку, Blockchain можна використовувати для формування уніфікованої рамки API [72].

Нещодавно було здійснено численні атаки на такі соціальні платформи, як Twitter і Facebook. Ці атаки призвели до злому даних, мільйони акаунтів були зламаними, а інформація користувачів потрапила в чужі руки. Технології Blockchain, якщо вони добре впроваджені в ці системи обміну повідомленнями, можуть запобігти таким майбутнім кібератакам, особливо якщо говорити про внутрішні структури та мережі держави, де це являється критичним фактором.

- Безпека IoT: хакери все частіше використовують периферійні пристрої, такі як термостати та маршрутизатори, щоб отримати доступ до загальних систем. Завдяки нинішній одержимості штучним інтелектом (ШІ), хакерам стало легше отримати доступ до загальних систем, таких як домашня автоматизація, за допомогою граничних пристроїв, таких як «розумні» перемикачі. У більшості випадків велика кількість таких пристроїв Інтернету речей має схематичні функції безпеки.

У цьому випадку Blockchain можна використовувати для захисту таких загальних систем або пристроїв шляхом децентралізації їх адміністрування. Підхід надасть можливості пристрою самостійно приймати рішення щодо безпеки. Незалежність від центрального адміністратора чи повноважень робить

периферійні пристрої більш безпечними, виявляючи підозрілі команди з невідомих мереж і діючи на них.

Зазвичай хакери проникають в центральне адміністрування пристрою і автоматично отримують повний контроль над пристроями та системами. Децентралізуючи такі системи повноважень пристроїв, Blockchain гарантує, що такі атаки важче виконувати (якщо це навіть можливо).

- Захист DNS і DDoS: Атака з розподіленою відмовою в обслуговуванні (DDoS) відбувається, коли користувачам цільового ресурсу, такого як мережевий ресурс, сервер або веб-сайт, відмовляють у доступі або обслуговуванні цільового ресурсу. Ці атаки вимикають або сповільнюють роботу ресурсних систем. З іншого боку, неушкоджена система доменних імен (DNS) дуже централізована, що робить її ідеальною мішенню для хакерів, які проникають у з'єднання між IP-адресою та назвою веб-сайту. Ця атака робить веб-сайт недоступним і навіть створюють перенаправлення на інші шахрайські веб-сайти. Blockchain можна використовувати для зменшення таких атак шляхом децентралізації записів DNS. Застосовуючи децентралізовані рішення, Blockchain видалив би вразливі окремі точки, якими користуються хакери.

- Децентралізація сховища. Злом даних і крадіжка стають основною очевидною причиною занепокоєння організацій. Більшість компаній досі використовують централізовану форму носія даних. Щоб отримати доступ до всіх даних, що зберігаються в цих системах, хакер просто використовує лише одну вразливу точку. Така атака залишає у розпорядженні злочинця конфіденційні дані, такі як фінансові записи бізнесу. Використовуючи Blockchain, конфіденційні дані можна захистити шляхом забезпечення децентралізованої форми зберігання даних. Цей метод зробить хакерам важчим і навіть неможливим проникнення в системи зберігання даних. Багато компаній, що користуються послугами зберігання даних, погоджуються, що Blockchain може захистити дані від хакерів. Apollo Currency Team є хорошим прикладом організації, яка вже прийняла технологію Blockchain у своїх системах (The Apollo Data Cloud).

- Програмне забезпечення: Blockchain можна використовувати для забезпечення цілісності завантаження програмного забезпечення, щоб запобігти сторонньому вторгненню. Так само, як використовуються хеші MD5, Blockchain можна застосовувати для перевірки діяльності, наприклад оновлення мікропрограми, інсталяторів і виправлень, щоб запобігти проникненню шкідливого програмного забезпечення на комп'ютери. У сценарії MD5 новий ідентифікатор програмного забезпечення порівнюється з хешами, доступними на веб-сайтах постачальників. Цей метод не є повністю надійним, оскільки хеші, доступні на платформі постачальника, вже можуть бути скомпрометовані. Однак у випадку з технологією Blockchain хеші постійно записуються в Blockchain. Інформація, записана в технології, не є змінною а отже, Blockchain може бути більш ефективним у перевірці цілісності програмного забезпечення, порівнюючи його з хешами в Blockchain.
- Перевірка кібер-фізичних інфраструктур: підробка даних, неправильна конфігурація системи разом із несправністю компонентів погіршили цілісність інформації, отриманої з кібер-фізичних систем. Однак можливості технології Blockchain щодо цілісності та перевірки інформації можуть бути використані для аутентифікації статусу будь-якої кіберфізичної інфраструктури. Інформація, отримана про компоненти інфраструктури через Blockchain, може бути більш надійною для повного ланцюга зберігання.
- Захист передачі даних. Blockchain можна використовувати в майбутньому для запобігання несанкціонованого доступу до даних під час їх передачі. Використовуючи повну функцію шифрування цієї технології, можна захистити передачу даних, щоб запобігти доступу зловмисників, будь то особа чи організація. Такий підхід приведе до загального підвищення впевненості та цілісності даних, що передаються через Blockchain. Хакери зі зловмисними намірами підтримують дані під час передачі, щоб змінити їх або повністю видалити їх існування. Це залишає величезний пробіл у неефективних каналах зв'язку, таких як електронна пошта.

Всі вищенаведені області застосування технології Blockchain можуть впроваджуватись як додатковий захист об’єктів критичної інфраструктури України, в мережі подвійного призначення, спеціальні інформаційні мережі, а також для захисту конфіденційних даних при передаванні їх відкритими мережами або мережами подвійного призначення [73]. Можна стверджувати, що в умовах гібридної війни або під час радіоелектронної боротьби, коли можливість передати військові, розвід дані, оперативні дані тощо є не можливим закритими мережами, то можна використати Blockchain технологію.

MaidSafe — аналогічна компанія, що базується у Великобританії. Їхня мета також полягає в децентралізації Інтернету та створення чогось на зразок альтернативного Інтернету, де користувачі зможуть запускати програми, зберігати дані та робити все інше, що вони зазвичай роблять онлайн, але в більш безпечному середовищі. Під час реєстрації в цій службі користувачі можуть вибрати, яку частину свого особистого місця для зберігання даних вони хочуть виділити для мережі. Потім система надає safecoin, криптовалюту, щоб компенсувати користувачам цінність (простір), яку вони пропонують мережі. Кожен файл, розміщений у мережі MaidSafe, шифрується, фрагментується та розповсюджується між користувачами. Єдина особа, яка може знову зробити дані доступними для читання, - це їх власник, забезпечуючи доступ до даних іншим особам, крім уповноваженого власника.

2.2. Безпечна та конфіденційна структура системи електронного урядування на основі Blockchain

Широка доступність Інтернету спонукала країни по всьому світу використовувати технології як засіб спілкування та обміну послугами між громадянами та іншими філіями. Користувачі електронного уряду насолоджуються онлайн-послугами, не виходячи зі своїх комфортних домівок, уникаючи довгих черг у державних установах, заощаджуючи час і транспортні витрати, і в той же час постачальники послуг можуть надавати послуги ефективніше та ефективніше. Загалом урядові мережі можуть спілкуватися одна з

одною краще, ніж бізнес-мережі, оскільки більшість із них підключено для передачі інформації громадськості без конкуренції. У майбутньому кількість пристроїв, які використовують послуги електронного уряду, різко зросте через швидку еволюцію розумних будинків, Інтернету речей (IoT), розумних міст та інших взаємопов'язаних мереж. Згідно з опитуванням ООН щодо електронного урядування, 2014 р., майже всі уряди в усьому світі зараз надають своїм громадянам та іншим зацікавленим сторонам електронні послуги через веб-сайти та мобільні додатки.

Системи електронного уряду збирають, зберігають і обробляють значну кількість конфіденційної інформації про громадян, співробітників, клієнтів, продукти, дослідження та фінансовий стан серед іншого за допомогою електронних комп'ютерів. Компрометація такої інформації зазвичай призводить до втрати довіри та впевненості користувачів, можливостей, фінансових переваг тощо. Було виявлено, що понад 80% веб-сайтів електронного уряду по всьому світу були вразливі до міжсайтових сценаріїв (XSS) і впровадження запитів (SQL) через відсутність належних механізмів автентифікації, застосованих до вхідних даних від користувачів. Останнім часом багато країн світу зіткнулися з великою загрозою атак на відмову в обслуговуванні (DoS) і шкідливих програм, націлених на їхні мережі. Наприклад, уряд США зазнав однієї з найбільших атак на електронний уряд у 2015 році, що призвело до витоку конфіденційної інформації понад 4 мільйонів державних службовців, включаючи інформацію про перевірку безпеки, номери соціального страхування, особи та паролі. Відповідно до звіту в 2016 році уряд Танзанії постраждав від кібертерористів, технологічних шпигунів, хакерів і цифрових шахраїв, що призвело до збитків у розмірі близько 85 мільйонів доларів США [74]. Крім того, у 2014 році було зламано понад 1500 облікових записів користувачів у Сінгапурі на урядовій платформі, де хакери отримали доступ для створення нових підприємств і подання заявок на отримання дозволів на роботу.

Тому дуже важливо забезпечити безпеку, приватність, конфіденційність, цілісність і доступність систем електронного урядування. Існуючі системи

електронного урядування, такі як веб-сайти електронного урядування та системи керування електронними ідентифікаторами, є централізованими, де один або дубльовані центральні сервери та бази даних зберігають та надають інформацію користувачам. Централізоване управління та система перевірки, ймовірно, страждатиме від єдиної точки збою, що робить систему мішенню для кібератак, таких як DDoS, DoS та іншого шкідливого програмного забезпечення. Будь-яка система електронного урядування залишатиметься вразливою до порушень конфіденційності та безпеки, якщо не буде розроблено та не буде доступно для боротьби з цими загрозами в майбутньому кращі технології безпеки та контрзаходи.

Технологія Blockchain виявилася одним з хороших рішень для забезпечення безпечного децентралізованого середовища для обміну інформацією. Хоча спочатку він був запроваджений для обміну цифрової валюти як базова технологія, він знайшов застосування безпеки та конфіденційності в багатьох інших сферах, таких як Інтернет речей (IoT), розумні будинки, розумні міста системи освіти та охорони здоров'я [75]. Хоча уряди в усьому світі не повністю запровадили технологію Blockchain у державному секторі, багато країн ініціювали проекти Blockchain, щоб дослідити потенціал технології Blockchain у пропонуванні державних послуг окремим особам. Кожен із цих проектів зазвичай зосереджується на конкретній послузі, наприклад, електронне проживання, електронна охорона здоров'я, земельний реєстр тощо; вони все ще перебувають на ранніх стадіях, і не було запропоновано жодної спільної основи для інтеграції технології Blockchain у системи електронного урядування. Крім того, кожна з цих країн розробляє власну структуру Blockchain. Різні системи Blockchain у різних країнах призводять до труднощів у спілкуванні за межами їхньої мережі для міжнародного обміну інформацією.

Система складається з однорангової мережі пристроїв (вузлів) електронного урядування та пристроїв користувачів. Коротко кажучи, будь-який новий пристрій електронного уряду або окремий пристрій, що приєднується до системи, буде перевірено існуючими одноранговими мережами, і один із однорангових

пристроїв буде обрано для встановлення мережевого вузла та Blockchain-адреси нового пристрою. Коли новий користувач намагається зареєструватися в системі через пристрій або один із державних відомств, користувачеві призначається ідентифікатор користувача та гаманець Blockchain для збору та зберігання його/її транзакцій. Завдяки цьому, користувачі електронного уряду можуть подавати та отримувати доступ до своїх записів з будь-якого місця та з будь-якого місця, використовуючи свої ідентифікатори та адреси Blockchain.

Послуги електронного уряду зазвичай можна класифікувати на 3 групи: уряд для уряду (УДУ), уряд для громадян (УДГ) і уряд для бізнесу (УДБ). УДУ пропонує онлайн-взаємодію між державними департаментами, органами влади, організаціями та іншими урядами для поширення інформації між собою за допомогою Інтернету. УДГ і УДБ дозволяють громадянам і підприємствам взаємодіяти з урядом, щоб отримати такі онлайн-послуги, як подання декларації про податок на майно, податок на прибуток, продовження/поновлення віз, паспортів і ліцензій, онлайн-голосування, подання заявок на електронні закупівлі тощо.

Кожен відділ електронного урядування гарантує, що лише авторизовані користувачі можуть отримати доступ до індивідуальної конфіденційної інформації. Окрім безпеки та конфіденційності, ще одним важливим фактором, який слід враховувати під час впровадження електронного уряду, є створення надійної системи, на яку користувачі можуть покластися. Гарантія безпеки та конфіденційності в системах електронного урядування відіграє вирішальну роль у збільшенні довіри між різними департаментами в уряді, оскільки це гарантує конфіденційність, цілісність і привілейований доступ до конфіденційної інформації. Типові атаки, з якими стикається система електронного урядування, включають сніфери пакетів, зонди, шкідливі програми, DDoS, фішинг тощо. Завдяки кібервійні з’явилися інші нові мотиви для атак, такі як політичні розбіжності, здирництво, кібертероризм і навіть змагання за перевагу, які можуть відбуватися всередині нації або між різними націями.

Були запропоновані різні нетехнічні моделі зрілості безпеки електронного уряду для керівництва та порівняльного аналізу впровадження безпеки системи електронного уряду. Ця модель зосереджена лише на налаштуванні механізмів безпеки організації, оцінці безпеки, налаштування політики безпеки та обізнаності користувачів про інформаційну безпеку, але в ній відсутні вказівки щодо вбудованої безпеки, яка може забезпечити безпеку та конфіденційність електронних послуг.

Як правило, безпека та конфіденційність систем електронного урядування забезпечуються за допомогою брандмауерів, систем виявлення вторгнень (IDS), інфраструктури відкритих ключів (PKI) та антивірусних механізмів. Різні методи штучного інтелекту були використані для реалізації IDS, які також можуть використовуватися системами електронного урядування. На додаток до цих звичайних рішень, одним із апаратних рішень проблем безпеки електронного уряду є використання системи eID для ідентифікації, автентифікації, конфіденційності та цілісності інформації користувачів. Система eID використовує смарт-картку, яка містить чіп, для зберігання персональних даних власників карток, включаючи дату народження, цивільний стан, батьківство, поточну та минулу адреси тощо, на додаток до сертифіката для автентифікації та цифрового підпису [76]. Система eID надає засоби для унікального розрізнення між різними громадянами та підприємствами для доступу до електронних послуг. У більшості країн міграційні служби та національні реєстри ідентифікаційних номерів пропонують eID-картки. Той самий eID можна використовувати в багатьох секторах (наприклад, оподаткування, соціальне забезпечення, освіта, послуги телефонії, банківські послуги), виконуючи різні ролі (наприклад, державного службовця чи бізнесмена) залежно від контексту. Існуючі системи управління eID не сумісні та не мають безпеки внаслідок наявності єдиної централізованої системи для ведення записів.

Іншим технічним рішенням для питань безпеки та конфіденційності в електронному уряді є структура автентифікації, запропонована в. У цій структурі використовуються різні процедури реєстрації та автентифікації на основі єдиного

центрального порталу для громадян, пов’язаного з міністерськими департаментами. Структура складається з двох частин, а саме: постачальник ідентифікаційної інформації і постачальник послуг. Користувачі повинні зареєструватися в IdP на центральному порталі системи електронного уряду, щоб отримати унікальні ідентифікатори, які використовуватимуться для доступу до послуг від постачальників послуг. Кожного разу, коли користувач хоче отримати доступ до служби, SP повинен підтвердити свою особу за допомогою IdP на центральному порталі, щоб отримати пароль єдиного входу (SSO), який можна використовувати для доступу до різних SP.

2.2.1. Розробка методу електронного урядування на основі Blockchain

Запропонована структура електронного урядування на основі Blockchain проілюстрована на рис. 2.2. На цьому малюнку двонаправлена стрілка УДУ показує взаємодію, що відбувається між державними департаментами та організаціями, що дозволяє здійснювати одноранговий (p2p) обмін (широкомовлення) та перевірку даних, які надаються особами. Подвійна стрілка УДГ позначає обмін інформацією між громадянами та урядом, як-от заповнення податкових форм, свідоцтв про шлюб, дозволів на ведення бізнесу, свідоцтв про народження, віз або паспортів. Подвійна стрілка УДБ вказує на обмін інформацією, як-от електронні закупівлі, податкові та страхові форми, а також електронні аукціони між державними та діловими організаціями (підприємствами) як головне джерело економічного зростання.

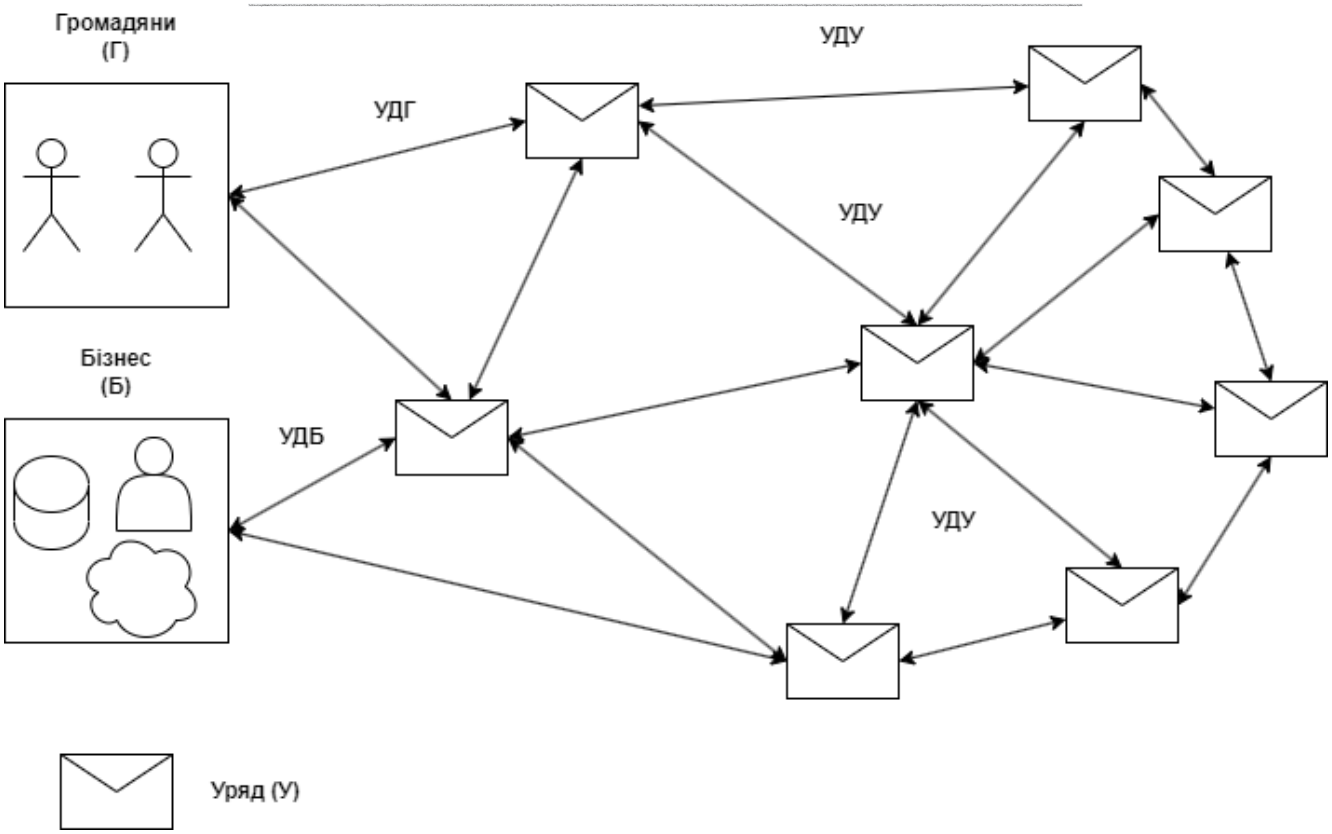

Рис. 2.2. Пропонований алгоритм роботи мережі електронного урядування на основі Blockchain

Приєднання будь-якого нового вузла електронного уряду (У) або вузла користувача (Г або Б) до мережі Blockchain перевіряється одноранговими користувачами в мережі, і токени електронного уряду призначаються для налаштування його мережевого вузла, що призводить до отримання дозволу (приватний) Blockchain. Кількість токенів електронного уряду еквівалентна загальній кількості записів, які зберігаються вузлом у мережі Blockchain. Кожен користувач має спеціальний гаманець електронного уряду для збору своїх жетонів. Після того, як запис буде подано, запис буде передано з точки зору частки на його/її адресу в Blockchain. Використовуючи протокол DPoS, будь-який вузол зможе зареєструватися в мережі як делегат. Щоб додати новий блок до Blockchain, департаменти, які спільно утворюють систему електронного уряду, голосують за делегата, який перевірить транзакції та запечатає блок. Цей підхід відповідає вимогам безпеки, оскільки випадкові вузли не можуть приєднуватися до мережі, генерувати нові маркери та налаштовувати мережевий вузол, якщо це

не схвалять інші вузли електронного уряду. Дозволена система Blockchain гарантує, що збережені записи є надійними, доступними для аудиту та прозорими.

2.2.2. Вузли мережі електронного урядування побудовані з використанням технології Blockchain

У термінології Blockchain існує два типи вузлів, які є повними вузлами та полегшеними вузлами. Повний вузол завантажує повну копію Blockchain, коли він приєднується до мережі Blockchain, що дозволяє йому повністю перевіряти транзакції та блоки [77]. Полегшений вузол не завантажує повну копію Blockchain, коли він приєднується до мережі, а завантажує лише заголовки блоків для перевірки автентичності транзакцій. Щоб мати можливість передавати свої транзакції в мережу та отримувати сповіщення, коли транзакції впливають на їхні гаманці Blockchain, полегшені вузли зазвичай посилаються на копію довіреного повного вузла Blockchain.

У запропонованій системі вузли відділу електронного урядування служать повними вузлами, тоді як пристрої користувача (У і Б) служать легкими вузлами, хоча будь-якому бізнес-вузлу дозволено завантажувати повну копію Blockchain. Підключення до мережі p2p у запропонованій мережі може бути забезпечено за допомогою бездротового широкосмугового зв’язку, завдяки тому факту, що багато країн у всьому світі намагаються об’єднати загальноміську бездротову широкосмугову мережу по всьому місту за допомогою технології Wi-Fi.

2.2.3. Голосування делегатів і свідків мережі Blockchain

Піри повинні узгодити стан транзакцій блоку та процес запечатування блоків у Blockchain, щоб мережа залишалася функціональною. DPoS прийнято тут як консенсусний алгоритм через його обчислювальну ефективність у додаванні транзакцій і запечатуванні блоку. Коротко, DPoS можна розглядати як представницьку демократію, де вузли електронного уряду використовують свою частку (записи), щоб обрати делегатів (інші вузли) для приєднання до мережі. Делегати відповідають за безпеку мережі та голосування за свідків серед них, які підтверджуватимуть транзакції та запечатуватимуть блоки.

Щоб обрати свідків, голоси зважуються відповідно до розміру жетонів кожного делегата або вузла голосування. Делегату не потрібно мати великий жетон, щоб бути обраним свідком, але голоси від делегатів із великими жетонами можуть призвести до того, що делегати з відносно маленькими жетонами будуть обрані свідками. Вузли електронного уряду зберігатимуть свої індивідуальні записи, що дозволить їм мати вищі маркери для участі та голосування за свідків, що підвищує безпеку запропонованої системи. У DPoS вузлам дозволено делегувати свої права голосу іншим вузлам, яким вони довіряють голосувати за свідків від їхнього імені [78]. Будь-який делегат, який, здається, поведе себе неправильно, буде вилучено з набору делегатів-учасників, а його голоси призначатимуться новому вузлу, який приєднується до мережі, або будь-якому існуючому вузлу, щоб забезпечити безпеку та довіру в мережі.

Вибір базової технології Blockchain для реалізації прототипу в основному залежить від доступності та ефективності реалізації DPoS. Тим часом обчислювальна енергія, необхідна для перевірки транзакцій, має бути доступною, а безпека створеної мережі має бути гарантована. Наприклад, платформа Ethereum реалізує DPoS і протокол смарт-контрактів для імітації реальних контрактів, таких як податкові та страхові контракти, трудові контракти та земельні реєстри. Він забезпечує важливу альтернативу в електронному урядуванні для зберігання конфіденційних записів громадян завдяки своїй здатності полегшувати переговори щодо контракту, спрощувати умови контракту, здійснювати виконання контракту та перевіряти стан виконання контракту.

2.2.4. Створення нового вузла мережі Blockchain для електронного урядування

Процес реєстрації нового вузла в запропонованій мережі електронного уряду підсумовано в «Алгоритмі 1». Будь-який відділ електронного урядування може приєднатися до мережі Blockchain, налаштувавши повний мережевий вузол, тоді як інші користувачі можуть налаштувати лише легкі вузли. Як тільки новий вузол приєднається до мережі, функціональний вузол згенерує свій гаманець Blockchain та адресу, що містить відкритий і закритий ключі, як показано в рядках

7 і 8 «Алгоритму 1». Закритий ключ використовується кожним вузлом для підписання та підтвердження транзакцій, тому його потрібно зберігати безпечно (рядок 9). Після генерації адреси вузол зв'яжеться з делегатами в мережі Blockchain, щоб надіслати свій запит на реєстрацію, і один із делегатів перевірить свою реєстрацію та передасть деякі токени електронного уряду (реєстраційний запис) на свою адресу Blockchain.

Після цього новий вузол додається до мережі, а його реєстрація транслюється одноранговим вузлам мережі призначеним делегатом (рядки 12–14), що дозволяє іншим одноранговим вузлам мережі отримувати інформацію про його гаманець для надсилання транзакцій у наступному циклі. Крім того, новий вузол отримує інструкції щодо налаштування мережевого вузла, щоб його можна було обрати делегатом для перевірки транзакцій у наступному циклі. Згодом новий вузол налаштовує вузол мережі відповідно до наданої інструкції. Зокрема, інструкція складається з розміру початкового токена, адреси Blockchain вузла, а також відкритого та закритого ключів для підписання та перевірки транзакцій перед додаванням блоку.

Процес додавання нового вузла завершується, коли мережевий вузол успішно налаштовано та передано в мережу делегатом. Інформаційна безпека підвищується завдяки використанню зашифрованих даних, які розповсюджуються по мережі. Таким чином, навіть якщо зловмисний вузол зареєстровано як вузол відділу, він не може змінити дані, оскільки кожен учасник мережі може виявити зміну та анулювати зміни.

Алгоритм 1: Додавання нової гілки в мережу елетронного уряду

1. $\boldsymbol{If}$ ($the\ request\ is\ from\ government$) $\boldsymbol{then}$
2. $Create\ a\ node\ n$;
3. $\boldsymbol{else}$
4. $Create\ a\ lightweight\ node\ n$;
5. $\boldsymbol{end\ if}$
6. $(K_{pub}, K_{pr}) \leftarrow generateKeys(\quad)$;
7. $Addr \leftarrow createBlockchainAdress() + (K_{pub}, K_{pr})$;

8. $Walt \leftarrow createBlockchainWallet() + (K_{pub}, K_{pr})$;
9. $safetyStorePrivateKey(\quad)$;
10. $Addr \leftarrow Addr + tokens$;
11. $\alpha \leftarrow selectDelegate\ (S)$;
12. $\boldsymbol{for\ each}\ m\ \in \{S - \alpha\}\ \boldsymbol{do}$
13. $distrReg\ (m, n)$;
14. $\boldsymbol{end\ for}$
15. $n \leftarrow verNewNODE(\quad)$;

2.2.5. Реєстрація користувача в Blockchain мережі для електронного урядування

Користувачі здійснюють реєстрацію за допомогою своїх мережевих пристроїв або фізично відвідуючи один із державних відомств, причому процес підсумовано в «Алгоритмі 2». Як описано в рядках 2 і 3 цього алгоритму, для користувача видається ідентифікатор, а для користувача генерується нова адреса Blockchain, що містить відкритий і закритий ключі, що дозволяє ідентифікувати власника. Гаманець Blockchain для цього нового користувача створюється та транслюється (рядки 5–8), щоб кожен вузол міг зберігати його у своїй адресі Blockchain. Потім створений гаманець Blockchain використовується для надсилання та отримання транзакцій, пов'язаних з обліковим записом користувача. Ідентифікатори користувачів і приватні ключі безпечно зберігатимуться у файлі гаманця або базі даних пристрою користувача. Користувачі можуть зручно переглядати свої записи та нові транзакції, доступні в їхніх Blockchain-адресах, через інтерфейс гаманця.

Алгоритм 2: реєстрація нового користувача

1. $(K_{pub}, K_{pr}) \leftarrow generateKeys(\quad)$;
2. $uID \leftarrow createUserID(\quad)$;
3. $Addr \leftarrow createBlockchainAdress() + (K_{pub}, K_{pr})$;
4. $(uID\ \ , K_{pr}) \leftarrow safetyStore(uID\ \ , K_{pr})$;
5. $Walt \leftarrow createBlockchainWallet() + (K_{pub}, K_{pr})$;

6. $\boldsymbol{for\ each}\ m \in S\ \boldsymbol{do}$
7. $distrWal\ (m, Wlt)$;
8. $\boldsymbol{end\ for}$
9. $a \leftarrow verNewUSER(\quad)$;

Коли користувач надсилає запис делегату, транзакція автентифікується та ініціалізується. Після цього блок оновлюється до нової версії, яка транслюється по всій мережі для перевірки, а потім передається на його/її адресу Blockchain в усіх однорангових мережах. Переданий запис зберігається в Blockchain-адресі користувача з таким вмістом даних: (1) ідентифікатор користувача, (2) значення запису, наприклад реєстрація власності, і (3) ідентифікація запису, наприклад номер податкової реєстрації . Кожен екземпляр даних у Blockchain представляє актив.

Коли стороння організація (наприклад, бізнес) запитує доступ до інформації користувача з будь-яких офіційних питань, користувач повинен надати свою адресу Blockchain для перевірки. Потім організація може використовувати веб-API Blockchain для доступу до даних Blockchain, що зберігаються в адресі користувача. Усі користувачі електронного уряду зобов'язані створювати резервні копії своїх особистих ключів і зберігати їх у безпеці. Якщо будь-який користувач втратив свій особистий ключ, він повинен буде створити нову адресу Blockchain та попросити один із вузлів відділу електронного урядування перенести його/її інформацію зі старої адреси Blockchain на новостворену адресу Blockchain. .

Коли зареєстрований користувач захоче отримати доступ до мережі, пристрій і ідентифікаційні дані користувача будуть перевірені та автентифіковані. Це допомагає звести до мінімуму людські помилки, які завжди вважалися основним внеском у збій і слабкою ланкою доступу до інформації, що зберігається в інформаційних системах. У результаті державна інформація буде безпечно надходити до потрібних осіб у потрібний час і в потрібному місці. Типові людські помилки в кібербезпеці включають надсилання конфіденційних даних не тому одержувачу та ненавмисне розкриття облікових даних для входу,

таких як імена користувачів і паролі. Людська помилка залишається однією з основних причин порушення кібербезпеки в державних і приватних організаціях.

2.2.6. Генерація нового блока мережі Blockchain для електронного урядування

Кожен блок створюється одним активним свідком, який вибирається випадковим чином більшістю однорангових користувачів зі списку активних делегатів. Якщо свідок пропускає блок, інший свідок отримує завдання створити та перевірити блок, щоб приєднатися до мережі Blockchain. У DPoS для створення блоку встановлюється фіксований період часу, часто п’ять секунд. «Алгоритм 3» висвітлює фундаментальні кроки, які беруть участь у процесі генерації та додавання блоку до Blockchain за допомогою алгоритму консенсусу DPoS.

Алгоритм 3: створення та додавання нового блоку в Blockchain

1. $initialize\ an\ empty\ set\ of\ transaction\ G = \{\ \ \}$;
2. $\alpha \leftarrow WitnesElect(S)$;
3. $\boldsymbol{while}\ treansaction_t < T_c\ \boldsymbol{do}$
4. $\boldsymbol{for\ each}\ m\ \in \{S - \alpha\}\ \boldsymbol{do}$
5. $G \leftarrow G + GetTransactionfromNode(m)$;
6. $\boldsymbol{end\ for}$
7. $\boldsymbol{end\ while}$
8. $b_{m+1} \leftarrow createBlock(b_m, G)$;
9. $\boldsymbol{for\ each}\ m\ \in \{S - \alpha\}\ \boldsymbol{do}$
10. $signBlock(b_{m+1}, m)$;
11. $\boldsymbol{end\ for}$
12. $B' \leftarrow B + b_{m+1}$;
13. $\boldsymbol{for\ each}\ m\ \in\ S\ \boldsymbol{do}$
14. $distributeBlockchain\ (B', m)$;
15. $\boldsymbol{end\ for}$

Блок додається через регулярний проміжок часу, Tc. У межах цього інтервалу блок проходить наступні фази діяльності. Спочатку ініціалізується

порожній набір транзакцій R; і один свідок із групи делегатів обирається для створення та перевірки транзакцій для Blockchain. По-друге, всі транзакції надсилаються обраному свідку. Цей процес триває, доки свідок не припинить приймати будь-які нові транзакції для блоку. По-третє, свідок збирає новий блок і розповсюджує його представникам мережі для перегляду та перевірки. Це дозволяє тим вузлам, які обрали свідка, цифровим підписом блоку підтвердити його правильність. Підписаний блок повертається свідку та додається до його локального Blockchain, одночасно розповсюджуючи новий блок у мережі. Свідок не може видобувати власну транзакцію, тому в рядках 4 і 9 b (свідок) виключається з набору вузлів N. Наприкінці алгоритму Blockchain поширюється на всі урядові вузли в мережі.

2.3. Взаємозв'язок між технологією Blockchain і ШІ, великими даними та іншими передовими технологіями

Розробка та застосування технології Blockchain були б неможливі без підтримки інфраструктури інформаційних технологій нового покоління, таких як ШІ, великі дані, хмарні обчислення та Інтернет речей [79]. У свою чергу, технологія Blockchain також сприяла розвитку цих інформаційних технологій. Очікується, що ШІ та Blockchain доповнюватимуть один одного відповідними перевагами.

1. ШІ може допомогти вирішити проблеми, з якими стикається Blockchain, щодо автономності, ефективності, енергоефективності та інтелекту, особливо достовірності даних у додатках ШІ, щоб ШІ міг більше зосередитися на алгоритмах. Крім того, штучний інтелект здатний більш ефективно керувати автономною організацією Blockchain, розширювати та покращувати функції та ефективність смарт-контрактів, а також оптимізувати операції Blockchain для підвищення безпеки, ефективності та енергоефективності.

2. Blockchain може забезпечити розподілений штучний інтелект, реалізувати взаємний виклик різноманітних функцій штучного інтелекту, прискорити розробку штучного інтелекту, порушити наразі закритий режим розробки та

сприяти обміну даними. Крім того, Blockchain також можна використовувати для моделей даних журналу аудиту, що забезпечує більш надійні прогнози тощо.

3. Коли кількість вузлів досягає певного масштабу, вартість атаки також буде величезною, що ускладнить реалізацію атаки. У цьому сенсі проблема витоку даних буде вирішена ефективно.

4. Вартість експлуатації та обслуговування IoT значно зменшується. Завдяки технології Blockchain IoT може передавати дані за принципом «точка-точка».

5. Пряме з'єднання без використання центральних процесорів. Розподілені обчислення можна використовувати для обробки сотень мільйонів транзакцій. Більше того, обчислювальна потужність, ємність зберігання та пропускна здатність сотень мільйонів неактивного обладнання значно зменшать витрати на обчислення та зберігання.

6. Перевірка третьою стороною не потрібна. Технологія Blockchain може допомогти вирішити проблеми масштабованості, одноточкового збою, відміток часу, запису, конфіденційності, довіри та надійності абсолютно послідовним чином. У повністю децентралізованій довіреній цифровій інфраструктурі обладнання IoT може працювати незалежно без необхідності отримання будь-якої централізованої авторизації.

7. Безпека IoT гарантована. Blockchain може записувати всі дії термінальних пристроїв. Оскільки записана інформація ніколи не може бути переписана, безпека даних і конфіденційність користувачів будуть поставлені під ефективний захист.

8. Послуги хмарних обчислень характеризуються великим масштабом, високою надійністю, низькою вартістю, гнучкістю та доставкою на вимогу. У поєднанні з децентралізацією та захистом даних від Blockchain він має потенціал для сприяння широкому застосуванню технології Blockchain. У майбутньому на основі інфраструктури та послуг (IaaS), платформи як послуги (PaaS) і програмного забезпечення як послуги (SaaS) буде створено Blockchain як послугу (BaaS) для інтеграції технології Blockchain в платформи хмарних обчислень. і

створити ринок хмарних послуг BaaS, таким чином забезпечуючи стабільні та надійні платформи хмарних обчислень для децентралізованих програм.

2.4. Перспективи військового застосування технології Blockchain

Командно-інформаційна система є важливим методом і обладнанням для реалізації точного й автоматичного командування на полі бою під час війни в реальному часі та допомагає штабам усіх рівнів здійснювати науково ефективне управління підлеглими підрозділами та озброєнням у звичайний час. Будучи органічною системою «людина-машина», в якій командири є ядром, а комп'ютери та використовують комп'ютери та інше інформаційно-технологічне обладнання як передумову та матеріальну гарантію, система органічно поєднує різні методи командування та управління та командирів, щоб забезпечити високу - рівень автоматизації збору, передачі, обробки та використання інформації військового управління. Таким чином, забезпечується цілісність даних, доступність і безпека командної інформаційної системи, а в основному гарантується підвищення боєздатності та успіх у захопленні ініціативи у війні, як зображено у таблиці 2.1.

Таблиця 2.1.

Рішення військових проблем з застосуванням технології Blockchain

Проблема	**Рішення**
Командна інформаційна система має централізовані мережі та бази даних	Використовуючи переваги децентралізованих і розподілених функцій, механізму консенсусу, автономних характеристик і кредитного механізму алгоритму асиметричного шифрування технології Blockchain, ми можемо створити автоматичну та безпечну систему командування та контролю.
Поточні БПЛА в кластерах не мають	використовуючи механізм консенсусу

загального сприйняття зовнішнього середовища, а також бракує ефективного обміну інформацією та координації дій між окремими БПЛА та формуваннями БПЛА.	технології Blockchain, ми можемо перетворити особисту довіру й інституційну довіру до машинної. На майбутньому полі бою система управління та управління кластеру БПЛА обмінюватиметься даними бойових команд у децентралізованій манері та таким чином уніфікує операції.

Інформаційна система Blockchain + Командування. По-перше, командна інформаційна система має централізовані мережі та бази даних. Будучи важливою ціллю у воєнний і навіть мирний час, він вразливий перед ворогами чи хакерами. Вони можуть використовувати різні інформаційні засоби для проведення мережевих і електричних атак, спричиняючи паралізацію всієї інформаційної системи, або для викрадення та фальсифікації ідентифікаційної інформації, підробки важливих даних. Тому учасники бойових дій стикаються з великими ризиками щодо достовірності даних і навіть можуть приймати помилкові рішення зі шкідливими даними [80]. По-друге, коли командири покладаються на командну інформаційну систему для прийняття рішень, існуватиме ризик віддавати неправильні команди чи фальшиві накази, спричинені підробкою даних ворогом. Технічний принцип: інформаційну систему команд, засновану на технології Blockchain, можна розділити на рівень даних, мережевий рівень, рівень консенсусу, рівень стимулювання, рівень контракту та рівень додатків. Серед них рівень даних гарантує надійність даних або розвідувальних даних, надійність і безпеку військової інформаційної системи; мережевий рівень гарантує самоорганізацію та децентралізацію функцій військових інформаційних систем; консенсусний рівень інкапсулює різні типи консенсусних алгоритмів для досягнення автономного та надійного прийняття рішень; рівень заохочення використовує програмований механізм заохочення, щоб уникнути всіх видів

неправильної поведінки та досягти стимулів позитивної поведінки; договірний рівень допомагає автоматизувати та інтелектуальні військові інформаційні системи, зменшуючи невизначеність, різноманітність і складність, яку люди та інші фактори привносять у бойове командування та військове управління.

Рішення: використовуючи переваги децентралізованих і розподілених функцій, механізму консенсусу, автономних характеристик і кредитного механізму алгоритму асиметричного шифрування технології Blockchain, ми можемо створити автоматичну та безпечну систему командування та контролю. Поєднавши його зі штучним інтелектом і військовим Інтернетом речей у майбутньому, ми можемо спочатку змінити бойовий контроль на тактичному рівні з централізованого режиму на децентралізований. Також можна досягти достовірного бойового командування, що означає, що команди, віддані командирами на всіх рівнях, можуть бути повністю записані, а переписані дані є доказами та простежуються. Лише коли ворог або хакери модифікують більше 51% інформації вузла одночасно (атака 51%), дані командної інформаційної системи можуть бути змінені. Завдяки даним часових рядів, спільному обслуговуванню та програмованим і надійним функціям технології Blockchain можна реалізувати динамічний і постійний запис у мережах або базах даних, а також можна ефективно запобігти неправдивій інформації та злому. Таким чином, ми можемо відстежувати в режимі реального часу, чи була підроблена база даних, чи відстежувалась військова система, і виключити ризик атак противника, притаманний командним інформаційним системам. Типовий випадок: DARPA намагається розробити безпечну та надійну інформаційну платформу на основі технології Blockchain. Це буде безпечна система інформаційного обслуговування, здатна ефективно захищати конфіденційні дані та запобігати злому.

Blockchain + операції кластера БПЛА. Кластерні системи БПЛА мають п’ять типових характеристик, а саме децентралізацію, автономне управління, відновлення кластера, посилення функцій і відсутність втрат. Однак поточні БПЛА в кластерах не мають загального сприйняття зовнішнього середовища, а

також бракує ефективного обміну інформацією та координації дій між окремими БПЛА та формуваннями БПЛА.

Рішення: використовуючи механізм консенсусу технології Blockchain, ми можемо перетворити особисту довіру й інституційну довіру до машинної. На майбутньому полі бою система управління та управління кластеру БПЛА обмінюватиметься даними бойових команд у децентралізованій манері та таким чином уніфікує операції. Здатність підтримувати боєздатність за будь-яких втрат має велике значення для реалізації нового командно-управлінського режиму «людина-машина/машина-машина», який відповідає безпілотним операціям. Перше стосується безпеки. Завдяки функціям «шифрування та дешифрування відкритих і закритих ключів» і «цифрового підпису» технології Blockchain кожен БПЛА в кластері може служити вузлом мережі. Усі вузли спільно використовують і ведуть ту саму ноду, забезпечують автентичність даних зв'язку та перевіряють ідентичність учасників кластера. Другий передбачає розподілене прийняття рішень. Розподілені алгоритми прийняття рішень є ключем до ефективної роботи кластерних систем БПЛА, а механізм консенсусу гарантує, що всі вузли в розподіленій системі узгоджують мету прийняття рішень [81]. У реальному бою кожна сутність у кластері повинна узгодити оперативні завдання та цілі, такі як групування та форматування, планування шляху та уникнення бар'єрів. Третє – для контролю формування. Технологія сайдчейн Blockchain дозволяє ієрархічно з'єднувати декілька Blockchain один з одним. З одного боку, БПЛА в різних ланцюгах можуть діяти відповідно до попередньо встановленого протоколу в ланцюжку, де вони розташовані; з іншого боку, міжланцюгова співпраця полегшує перемикання між різноманітними кластерними утвореннями. У процесі скоординованого пошуку, розвідки та атаки кластер БПЛА може змінювати стрій, щоб захистити себе та знищити ворога.

Четвертий – про децентралізовані автономні кластери. У майбутньому кожна сутність у кластеры БПЛА розглядатиметься як автономний агент із функцією сприйняття, міркування та прийняття рішень. Ці агенти формуватимуть

різні децентралізовані автономні кластери за допомогою смарт-контрактів для автономного виконання оптимальних рішень.

2.4.1. Автономне управління БПЛА в режимі “рою”

У сучасних системах боротьби з терористами безпілотники активно розгортаються в польових операціях для спостереження, розвідки та цілеспрямованих атак. Також безпілотники можуть відігравати важливу роль під час рятувальних операцій у зонах, що постраждали від катастроф і природних катаклізмів. Наприклад, дрони доставляють аптечки першої допомоги постраждалим до того, як бригада медиків прибуде на місце.

Система безпілотника включає в себе вбудовані датчики, GPS, стабілізатори та камери високої чіткості для збору та розповсюдження даних для представлення детальної обізнаності про ситуацію на наземному диспетчері. Контролер може наказувати дронам змінювати їх поведінку або фізичне положення. У відповідь він коригує, збирає та пересилає оновлені дані, щоб представити ситуацію диспетчеру. Для порівняння, автономні безпілотники можуть відстежувати свою швидкість, рух і ресурси для незалежного реконфігурації та перенастроювання для досягнення своєї мети. На продуктивність безпілотників впливає наявність супротивників. Наприклад, перешкоди комунікації супротивників між безпілотником і контролером можуть призвести до провалу місії безпілотника. Змагальні вузли у багатьох випадках вставляють регулятори у вбудоване програмне забезпечення дронів, щоб зламати систему.

Рій складається з кількох автономних дронів, які працюють разом для досягнення мети. Кожен дрон у групі може злітати, приземлятися та зависати. Дрони організовані в шари, які називаються кластерами, і один із дронів обирається лідером рою для зв’язку з наземним диспетчером. Кожен дрон може спілкуватися з лідером зграї та дронами того самого кластера, щоб обмінюватися інформацією. Зв'язок між дронами допомагає безпілотникам коригувати свою поведінку у відповідь на дані в реальному часі. Однак рій вразливий до військових атак, оскільки вороги можуть перехопити зв’язок між дронами, щоб порушити дані.

Керування роями за допомогою Blockchain може допомогти зареєструвати кожен безпілотник на платформі Blockchain. Транзакції з цифровим підписом, походження даних і механізм консенсусу можуть допомогти негайно ідентифікувати та звести нанівець дані, які були пошкоджені між дронами. Blockchain-рішення на основі смарт-контрактів із контролем доступу гарантують, що лише зареєстровані вузли можуть отримати доступ до ноди для участі в ройових операціях.

В оборонному секторі рій може ідентифікувати потенційні цілі, середовище та небезпеки на основі даних у реальному часі. Кожен дрон зграї однаково бере участь у розпізнаванні ситуації чи цілі.

Технологію Blockchain можна використовувати для досягнення прозорого та високонадійного глобального прийняття рішень роєм, яке можна перевірити. Наприклад, безпечне голосування за допомогою Blockchain може допомогти зграї дронів прийняти глобальне рішення. У системах військового захисту зграї можна використовувати для (а) виявлення ворогів, (б) ідентифікації моделей пересування, (в) розпізнавання поранених солдатів на полі бою, (г) визначення перешкод і (д) каталогізації жертв у надзвичайних ситуаціях. На рисунку 1 представлена методологія з використанням Blockchain для виявлення зловмисників на полі бою. Камера дрона лідера зграї знімає відео для розпізнавання об’єктів на полі бою. У відповідь лідер створює дві адреси (розумні контракти) на Blockchain, щоб запросити членів рою дізнатися їхню думку. У відповідь кожен зареєстрований дрон запускає свій алгоритм розпізнавання об’єктів і оцінює результат. Він розраховує ймовірність того, що об’єкт є зловмисником, і записує результат у Blockchain. Розумний контракт, що зберігається в Blockchain, автентифікує та перевіряє походження даних. Нарешті, смарт-контракт об’єднує думки всіх безпілотників, щоб опублікувати кінцевий результат.

Катастрофи та стихійні лиха, такі як землетруси, можуть пошкодити мережеву інфраструктуру, що призводить до обмеження зв’язку між рятувальними командами.

Відповідно, це впливає на роботу рятувальної групи. Типові системи забезпечують безперервний зв’язок рятувальних команд, таких як пожежні бригади, військові, поліція, медичний персонал і організації соціального забезпечення, щоб обмінюватися даними. Безінфраструктурні спеціальні мережеві технології використовуються для з’єднання рятувальних команд у постраждалих від стихійних лих районах. Захищений і контрольований рій можна використовувати для дистанційного моніторингу рятувальних операцій у певній місцевості. Ройові послуги можуть включати спостереження, зондування та логістику для постраждалих районів. Рій може складатися з різнорідних дронів, якими володіють і керують різні партнери.

Процес реєстрації для схеми рою за допомогою Blockchain реєструє кожного власника та дрона в мережі Blockchain. Користувач може відстежувати деталі роя, такі як тип послуги, вартість і репутація в ланцюгу, щоб найняти його (через відстеження незмінних записів). На наступному етапі користувач вносить криптовалюту в гаманець смарт-контрактів, щоб використовувати сервіси роя. Сертифікат угоди про рівень обслуговування (SLA) має цифровий підпис і зберігається на сервері IPFS. Хеші сертифіката реєструються в Blockchain. Рій надає обрані послуги користувачам протягом певного періоду. Усі транзакції та дані з цифровим підписом незмінно записуються в книгу Blockchain для цілей аудиту. Після надання послуги смарт-контракт перераховує суму на гаманець власника для здійснення платежу (підлягає перевірці). Таким чином, технологію Blockchain можна використовувати для створення надійної системи серед ненадійних користувачів.

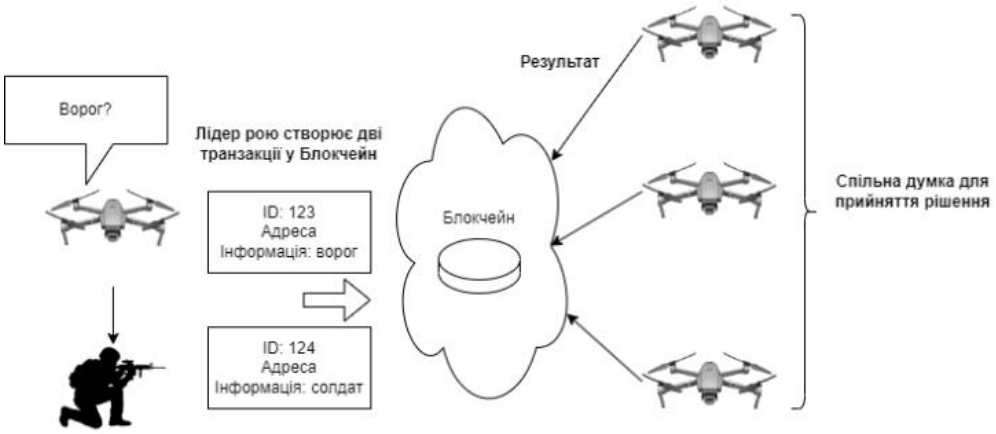

Рис. 2.3. Методологія з використанням Blockchain для виявлення зловмисників на полі бою.

2.4.2. Військове управління з використанням технології побудованих на основі Blockchain

Традиційна структура військового управління — це система «командування та контролю» зверху вниз, яка породжує такі проблеми, як роздуті інституції, високі витрати на управління, нечітке визначення відповідальності, неефективне управління, надмірне управління, погана передача інформації та концентрація влади на верхнього рівня, тоді як нижні рівні мають дуже обмежену автономію і, отже, обмежений інноваційний потенціал.

Blockchain + управління зброєю. Сучасна війна вимагає кращої чутливості та спритності бойових систем, але інститути, що мають ієрархічну структуру працює неефективно. Необхідно записати велику кількість даних, таких як проектні плани, результати випробувань і стан бойової техніки протягом усього життєвого циклу зброї, починаючи з проекту, демонстрації, розробки, виробництва та надання послуг до виходу на пенсію. Інформацію легко втратити або підробити в процесі.

Рішення: на основі технології Blockchain ми можемо побудувати систему управління зброєю та обладнанням протягом життєвого циклу, якою спільно керують і взаємно контролюють розробники, виробники та користувачі. За допомогою системи ми можемо відстежувати та керувати параметрами конструкції обладнання, даними випробувань, станом бойової техніки та

записами про технічне обслуговування. Жодним вмістом не можна маніпулювати чи видаляти, що покращує інформаційну безпеку, зручність і надійність. Усі дії управління покладатимуться на розумні контракти для відкритих і прозорих рішень, зменшення ієрархії управління, формування плоского режиму управління та, нарешті, підвищення ефективності. У той же час можна простежити походження кожного компонента озброєння та обладнання, що допомагає вирішувати суперечки щодо контрактів на закупівлю та створює повну непорушну систему моніторингу, управління та контролю, таким чином покращуючи безпеку управління, зручність та довіру.

Blockchain + управління конфіденційними даними. Використовуючи динамічну функцію технології Blockchain, ми можемо забезпечити вирішення проблеми «важко підтримувати докази» в управлінні конфіденційними даними у військовій інспекції та нагляді, людських ресурсах, медицині та охороні здоров'я. «Правдивий запис» усієї інформації можна реалізувати за допомогою «свідка всієї мережі», що дозволяє уникнути підробок документів, втрат файлів і фальсифікації інформації.

Військова логістика. Важко вирішити проблеми, пов'язані з мережею, зберіганням даних, обслуговуванням системи, відстежуваністю та контролем якості під час пакування, завантаження та розвантаження, транспортування та розбирання у військовій логістиці. Рішення: використовуючи технологію Blockchain, ми можемо створити окремий, безпечний, спільний і постійний запис, який можна перевірити, відстежувати та перевіряти транзакції різних ланцюгів постачання та між усіма операційними партнерами, а також здійснювати ефективне управління життєвим циклом ланцюга поставок оборонного призначення та системи закупівлі та логістика. Впровадивши технологію Blockchain у військову логістичну мережу, ми можемо побудувати децентралізовану автономну мережу персоналу та матеріалів у логістичних системах. Дані, пов'язані з виробництвом, закупівлею, транспортуванням і розподілом матеріалів у системах, можуть зберігатися в різних блоках в уніфікований спосіб, що може значно підвищити безпеку військової логістичної

інформації та дозволити військовому матеріально-технічному обладнанню та енергії відповідати вимогам. різних служб і відділів, тому завжди в найкращому стані.

Сучасна військова логістика передбачає розумне складування, пакування, транспортування та розподіл. Різні процеси утворюють невеликий військовий IoT на основі динамічної автономної мережі людей і об'єктів. Використовуючи вузли, ми можемо спілкуватися безпосередньо або через ретранслятори, щоб керувати важливими даними у військовому логістичному ланцюгу, такими як потреби користувачів, товари, що зберігаються, завантаження, транспортування, розподіл і транзит. Технічне обслуговування Blockchain контролюватиметься всіма вузлами по всій мережі, а незаконна робота деяких окремих вузлів буде відхилятися та протистояти більшості вузлів. Таким чином підвищиться безпека та зручність транзакцій, а також скоротиться час, який витрачається на інтелектуальну військову логістику. Він також може допомогти вирішити проблеми, пов'язані з мережею, зв'язком, зберіганням даних і обслуговуванням системи під час пакування, завантаження, розвантаження, транспортування та розбирання у військовій логістиці, таким чином забезпечуючи впорядковану та ефективну роботу системи. Це вважається найбільш перспективним військовим застосуванням Blockchain. Типовий випадок: у квітні 2016 року Міністерство оборони США та його союзники по НАТО почали звертати увагу на потенційне застосування технології Blockchain в обороні, включаючи автоматичне виконання смарт-контрактів, безпечне зберігання конфіденційних файлів і зменшення помилок і перерв. під час виконання оборонного контракту. Крім того, технологію Blockchain також можна застосувати для реагування на надзвичайні ситуації, коли трапляються катастрофи, і підвищити прозорість закупівель сировини в ланцюжку поставок і транспортування барж у процесі логістики. У червні 2017 року ВМС США скористалися технологією Blockchain для підвищення безпеки систем адитивного виробництва. Вони записували весь процес проектування компонентів, виготовлення прототипу, тестування, виробництва та остаточної

обробки, щоб користувачі могли переглядати будь-які конкретні дані та повідомляти про пошкодження компонентів або в кінці їх життєвого циклу.

2.4.3. Військова безпека Blockchain мережі

Квантовий Blockchain. Необхідно забезпечити безпеку передачі даних на полі бою, вирішити проблеми перехоплення, дешифрування та розвідки сигналів.

Рішення: використовуючи квантовий розподіл ключів як заміну оригінальної структури закритого ключа та використовуючи функції квантової криптографії Blockchain-мережі для захисту від підслуховування та перехоплення, ми можемо значно підвищити захисну здатність мережі Blockchain і справляти руйнівний вплив на проблему перехоплення, дешифрування та виявлення сигналу.

Blockchain + еластична комунікація. Має бути забезпечена безпека передачі даних на полі бою. Рішення: ґрунтуючись на розподілених характеристиках технології Blockchain, можна побудувати систему зв’язку із широким охопленням, стійку до аварій та високий рівень безпеки, щоб досягти безпечного та стійкого зв’язку в складному середовищі на полі бою. Типовий випадок: у травні 2017 року компанія Technology and Manufacturing Company зі штату Індіана, фінансована DARPA, використала технологію Blockchain для розробки «недоступної платформи обміну повідомленнями та торгівлі» для військових. Ця платформа розділяє створення та передачу інформації, щоб гарантувати, що надіслані та отримані дані неможливо зламати, і забезпечити безпечний зв’язок між штабом і наземними силами, а також між Міністерством оборони та офіційними особами розвідки. У липні 2019 року в рамках стратегії цифрової модернізації Міністерства оборони DARPA почало використовувати технологію Blockchain для створення ефективнішої, потужнішої та безпечнішої комунікаційної платформи, щоб сприяти безпечній передачі інформації для будь-яких командних, контрольних і комунікаційних систем, що дозволяють персонал для відстеження транзакцій через канал розподіленої книги та гарантування безпеки зв’язку між

штабом і сухопутними військами, а також між офіційними особами Міністерства оборони та розвідки в майбутньому.

2.5. Аналіз безпеки та конфіденційності

Кожна система електронного урядування має гарантувати конфіденційність, цілісність і доступність послуг. Конфіденційність досягається, коли інформація не розкривається неавторизованим користувачам; Цілісність досягається шляхом захисту інформації від будь-якої форми модифікації, тоді як доступність означає, що інформація доступна в разі потреби та вільна від DoS або DDoS чи інших подібних порушень служби. У цьому розділі представлено теоретичний якісний аналіз ефективності безпеки та конфіденційності системи електронного урядування на основі Blockchain.

Записи, що зберігаються в запропонованій системі, захищені криптографією з відкритим ключем, яка захищає від зловмисних спроб змінити та/або неавторизованого доступу, тоді як користувачам мережі призначаються закриті ключі для підтвердження та підписання транзакцій. Шифрування та цифровий підпис використовуються в мережі для забезпечення безпеки, конфіденційності та контролю доступу до збережених записів. Крім того, більшість консенсусних алгоритмів Blockchain (у цьому випадку DPoS) вимагають від зловмисника контролю принаймні 51% однорангових мереж, щоб змінити запис, чого, як правило, неможливо досягти. Точніше, щоб змінити будь-який блок у ланцюжку блоків, зловмисник повинен змінити кожну копію цього блоку в мережі, а потім переконати більшість вузлів, що новий блок є дійсним. Крім того, для підвищення конфіденційності даних, що зберігаються в запропонованій мережі, усі блоки користувачів хешуються, а незрозумілі хеші транзакцій зберігаються в Blockchain.

Запропонована система є децентралізованою системою p2p, де дані користувача зберігаються в різних вузлах, що гарантує доступність системи, уникаючи будь-якої єдиної точки збою. Використовуючи DPoS, будь-якому зловмиснику важко запускати DDoS або DoS атаки проти системи, оскільки потрібна реєстрація для того, щоб вузол почав обмінюватися інформацією з

рештою однорангових мереж. Будь-які транзакції, отримані від вузла мережі, перевіряються свідками, що ускладнює зловмисним вузлам ініціювання зловмисних з'єднань.

Послуги безпеки та відповідні загальні заходи, які надає запропонована структура, підсумовані в таблиці 2.2, що забезпечує адекватну конфіденційність і безпеку транзакцій. Для підвищення обчислювальної ефективності пристрої користувача запускатимуть полегшені клієнти для зберігання транзакцій, а не повну копію Blockchain, яка є дорогою з точки зору зберігання. Очікується, що пристрої електронного уряду будуть обчислювально потужними з достатньою ємністю для зберігання та ефективної обробки записів користувачів. Мережа здатна запропонувати продуктивність, яку забезпечують технологія Blockchain і консенсусний протокол DPoS, наприклад масштабованість, швидкість, сумісність і прозорість, і вона може обробляти велику кількість транзакцій.

Таблиця 2.2.

Служби безпеки та загальні заходи

Служба безпеки	Протидія (заходи)
Автентифікація	Blockchain-адреса та цифровий підпис
Контроль доступу	Цифровий підпис і шифрування
Конфіденційність	Шифрування
Цілісність	Шифрування та цифровий підпис
Невідмовність	Шифрування та цифровий підпис
Доступність	Розподілений/децентралізований
Довіра	Децентралізація, шифрування та цифровий підпис

Підхід криптографії еліптичної кривої (ECC) використовується для реалізації шифрування та цифрового підпису в запропонованій структурі, що є загальною практикою для більшості існуючих технологій Blockchain, таких як Bitcoin та Ethereum. Зверніть увагу, що ECC і RSA (Rivest-Shamir-Adleman) пропонують подібний рівень безпеки, але ECC споживає набагато меншу

кількість бітів. Наприклад, 256-бітний ключ у ECC пропонує такий самий рівень безпеки, як і RSA, що використовує 3072-бітний ключ. Коротший ключ зазвичай означає низьке споживання процесора, низьке використання пам’яті та швидке створення ключа. Ці переваги також сприяють швидкому створенню транзакцій і запечатуванню блоків. Короткий виклад дослідження довжини ключа між RSA та ECC наведено в таблиці 2.3. 256-бітні ключі ECC широко поширені в технології Blockchain, оскільки вони можуть забезпечити необхідний рівень безпеки для більшості програм.

Таблиця 2.3.

Порівняння довжин ключів RSA та ECC у бітах

Довжина ключа RSA	Довжина ключа ECC	Прибл. співвідношення (RSA:ECC)
1024	160	6:1
2048	224	9:1
3072	256	12:1
7680	348	20:1
15360	512	30:1

Окрім безпеки та збереження конфіденційності, система електронного урядування на основі Blockchain також забезпечує низку інших переваг, підсумованих у таблиці 2.4. Ці функції роблять технологію Blockchain перспективним напрямом у впровадженні системи електронного урядування, яка може забезпечити зручний, безпечний та відмовостійкий канал зв’язку між державним сектором та громадянами. Непрямі переваги, які приносять технології Blockchain, такі як скорочення бюрократії, виключення використання паперу, скорочення транзакційних витрат і контроль корупції, можуть змінити екосистему управління з вищим ступенем довіри з боку громадян.

Таблиця 2.4.

Особливості системи електронного урядування на базі Blockchain

Особливість	Пояснення
Зменшення людських помилок	Пристрої та ідентифікаційні дані користувачів проходять автентифікацію заздалегідь перед отриманням доступу до мережі
Підвищення суспільної довіри	Люди мають прямий контроль над своєю інформацією, і всі учасники мережі автентифіковані
Більша масштабованість	Систему можна легко масштабувати, оскільки вона дозволяє автоматично додавати нові пристрої та користувачів до мережі відповідно до механізму консенсусу
Підвищена надійність	Дані зберігаються на кількох серверах/розташуваннях. Протокол консенсусу гарантує, що дані можуть бути змінені лише за згодою всіх учасників
Підвищена відмовостійкість	Уникає єдиної точки збою, а система, отже, стійка до зловмисного програмного забезпечення, атак DoS і DDoS
Покращена можливість перевірки	Легко відстежити історію всіх транзакцій, оскільки вони залишаються незмінними в мережі
Краща можливість перевірки	Усі нові транзакції перевіряються всіма учасниками мережі перед додаванням до Blockchain
Право власності на інформацію	Особи несуть відповідальність за надання доступу до їх інформації
Покращений доступ до інформації	Інформація зберігається в кількох місцях, що покращує легкий і швидкий доступ
Підвищена якість даних	Усі транзакції та записи, що зберігаються в системі, перевіряються заздалегідь, що робить збережену інформацію автентичною з необхідною якістю
Більша прозорість	Усі вузли в мережі спільно використовують ту саму копію

	Blockchain, а нові транзакції додаються на основі механізму консенсусу
Зниження експлуатаційних витрат	Для обробки транзакцій сторонні організації не потрібні
Покращена ефективність і швидкість	Будь-хто в мережі може отримати доступ до всіх записів, що підпадають під привілей доступності, і нові записи поширюються на всі вузли-учасники

2.6. Переваги та недоліки використання технології Blockchain для захисту кіберсистем

Технологічна складність побудови Blockchain для захисту кіберсистем викликає певні занепокоєння щодо впровадження, безпеки та стійкості. Давайте детальніше розглянемо плюси та мінуси Blockchain у контексті кібербезпеки та захисту даних.

Основними перевагами технології Blockchain для кібербезпеки є наступне:

- Децентралізація - завдяки одноранговій мережі немає необхідності в перевірці третьої сторони, оскільки будь-який користувач може бачити мережеві транзакції.
- Відстеження - усі транзакції в Blockchain мають цифровий підпис і штамп часу, тому користувачі мережі можуть легко відстежувати історію транзакцій та відстежувати облікові записи в будь-який історичний момент. Ця функція також дозволяє компанії мати достовірну інформацію про активи або розподіл продукції.
- Конфіденційність - конфіденційність учасників мережі висока завдяки криптографії з відкритим ключем, яка автентифікує користувачів і шифрує їхні транзакції. Однак деякі стартапи на основі Blockchain йдуть ще далі й удосконалюють цю технологію. Наприклад, Guardtime розробила інфраструктуру безключового підпису (KSI), яка дозволяє користувачам перевіряти дійсність свого підпису, не розкриваючи ключі.

- Право бути забутим - конфіденційність даних важлива, навіть якщо ваша інформація незмінна. Оскільки немає можливості стерти непотрібну інформацію, технологія Blockchain забезпечує конфіденційність ваших даних, коли ви забуваєте ключ, оскільки ніхто не може його розшифрувати.
- Безпека від шахрайства - у разі злому легко визначити зловмисну поведінку завдяки одноранговому з'єднанню та розподіленому консенсусу. На сьогоднішній день Blockchain вважаються «незломними», оскільки зловмисники можуть вплинути на мережу, лише отримавши контроль над 51% мережевих вузлів.
- Сталість - технологія Blockchain не має єдиної точки збою, а це означає, що навіть у разі DDoS-атак система буде працювати в звичайному режимі завдяки кільком копіям ланцюга.
- Цілісність - розподілений реєстр забезпечує захист даних від модифікації або знищення. Крім того, технологія забезпечує достовірність і незворотність здійснених транзакцій. Зашифровані блоки містять незмінні дані, стійкі до злому.
- Стійкість - одноранговий характер технології гарантує, що мережа буде працювати цілодобово, навіть якщо деякі вузли знаходяться в автономному режимі або піддаються атаці. У разі атаки компанія може зробити певні вузли резервними і працювати як зазвичай.
- Якість даних - технологія Blockchain не може покращити якість ваших даних, але вона може гарантувати точність і якість даних після їх шифрування в Blockchain.
- Захищений доступ до мережі - співробітники можуть мати потребу в постійному доступі до Blockchain з кількох пристроїв, тому компанія ризикує втратити контроль над своїми закритими ключами. Щоб уникнути ризиків, пов'язаних із втратою ключів або людською помилкою, Blockchain REMME надає кожному користувачеві та кожному пристрою спеціальний сертифікат рівня Secure Sockets Layer, який усуває потребу в паролях. Такий підхід унеможливлює несанкціонований доступ до мережі.
- Захищене спілкування - ділове листування містить конфіденційні дані, які можна ефективно захистити, якщо ви використовуєте Blockchain для

кібербезпеки. Є багато стартапів, які шифрують ділове спілкування. Наприклад, Obsidian використовує мережі на основі Blockchain, щоб пом’якшити вразливості в наскрізному шифруванні. Розподілений ланцюг для повідомлень знижує ризик спостереження.

- Розумні контракти - програми, які базуються на реєстрі. Ці програми забезпечують виконання умов контракту та перевіряють сторони. Технологія Blockchain може значно підвищити стандарти безпеки для смарт-контрактів, оскільки мінімізує ризики кібератак і помилок.
- Доступність - немає необхідності зберігати ваші конфіденційні дані в одному місці, оскільки технологія Blockchain дозволяє вам мати кілька копій ваших даних, які завжди доступні для користувачів мережі.
- Підвищення рівня довіри клієнтів - клієнти будуть більше довіряти вам, якщо ви зможете забезпечити високий рівень безпеки даних. Крім того, технологія Blockchain дозволяє миттєво надавати своїм клієнтам інформацію про продукти та послуги.

Хоча Blockchain змінює кібербезпеку, все ще є деякі недоліки, які слід враховувати:

- Необоротність - існує ризик того, що зашифровані дані неможливо буде відновити, якщо користувач втратить або забуде закритий ключ, необхідний для їх розшифрування.
- Обмеження зберігання (місця) - кожен блок може містити не більше 1 Мб даних, а Blockchain може обробляти лише 7 транзакцій в секунду.
- Ризик кібератак - хоча ця технологія значно знижує ризик зловмисного втручання, вона все ще не є панацеєю від усіх кіберзагроз. Якщо зловмисникам вдасться захопити більшу частину вашої мережі, ви можете втратити всю базу даних.
- Проблеми з адаптацією - хоча технологію Blockchain можна застосувати практично до будь-якого бізнесу, компанії можуть зіткнутися з труднощами при її інтеграції. Застосовувати цю технологію в системах ланцюга поставок, наприклад, досить складно, оскільки може знадобитися багато часу, щоб відтворити ланцюги

поставок у вигляді Blockchain і вдосконалити їх. Blockchain-додатки також можуть вимагати повної заміни існуючих систем, тому компанії повинні розглянути це перед впровадженням технології Blockchain.

- Високі експлуатаційні витрати - запуск технології Blockchain вимагає значної обчислювальної потужності, що може призвести до високих граничних витрат у порівнянні з існуючими системами.
- Blockchain грамотність - досі не вистачає розробників з досвідом роботи в технології Blockchain і глибокими знаннями криптографії.

Загалом, технологія Blockchain є проривом у кібербезпеці, оскільки вона може забезпечити найвищий рівень конфіденційності, доступності та безпеки даних. Однак складність технології може викликати труднощі з розробкою та використанням у реальному світі. Технологія Blockchain спирається на останні криптографічні досягнення, а також на всебічний досвід управління мережею.

Висновки до розділу 2

Таким чином, у другому розділі дисертаційної роботи було проведено дослідження використання систем побудованих на основі Blockchain у різних інформаційних сферах, зокрема для побудови системи урядового управління та для покращення систем військовго управління. Окрім цього проведене дослідження дозволяє виділити наступне:

1. Технологія Blockchain протягом років показала себе стійкою до будь-яких атак. Нею користуються для підтримання безпеки однієї з найчутливіших сфер – фінансової. Однак саму технологію та її властивості поступово починають використовувати й в багатьох інших сферах також. Використання цієї технології для побудови захищених комп’ютерних мереж, якими користувачі будуть користуватися повсякденно для персонального користування чи для користування в середині організації чи державної структури – це питання часу та досліджень. Побудова та впровадження захисних механізмів, визнаних у всьому світі буде сприяти розвитку технологій та дозволить почати процес впровадження нової версії комп’ютерних мереж в масове користування.

2. Аналіз сучасного стану IoT показує, шо оскільки все більше людей приєднується до всесвітньої мережі, а технології продовжують розвиватися, з’являється все більше даних, і все більше зловмисників які намагатимуться вкрасти або зіпсувати ці дані. Технологія, що лежить в основі Blockchain, є універсальною та неймовірно корисною для майбутнього Інтернету, дозволяючи користувачам краще захищати свої дані.

3. Дослідження та аналіз систем захисту, що побудовані на основі Blockchain показали, що дана система має свої переваги та недоліки (в ціні побудови такої системи, спеціалісти в даній області зараз дуже дорогі), однак переваги звісно переважають. Особливо якщо говорити про таке актуальне питання як військова справа. Використання Blockchain технології дозволить покращити процес децентралізації управління військами особливо в таких сферах як управління дронами. В наш час сучасна війна перетворюється у війну дронами. На разі ними керують оператори, але використання Blockchain технології в парі з ШІ (штучним інтелектом) дозволить створювати системи “Роїв”, які не потребуватимуть централізованого управління оператором, що дозволить набагато ефективніше комунікувати їхні дії та вибір цілей за рахунок використання системи “свій-чужий.

4. Програмні приманки, як технологія, може отримати новий етап розвитку, якщо її функції об’єднати з динамічними атрибутами технології Blockchain. Вони можуть підсилити захист комп’ютерної мережні, побудованої на основі даної технології та стати новим рівнем захисту, моніторингу та попереджень атак зовнішнішні так і внутрішніх.

РОЗДІЛ 3. РОЗРОБКА МЕТОДУ ВИКОРИСТАННЯ ПРОГРАМНИХ ПРИМАНОК ЯК ЕЛЕМЕНТІВ ЗАХИСТУ КОМП'ЮТЕРНИХ МЕРЕЖ НА ОСНОВІ ТЕХНОЛОГІЇ BLOCKCHAIN

3.1. Розробка моделі децентралізованого звязку на основі приманок

Аналіз систем обману (Deception) та Honeypot в розділі 1 показав перспективи розвитку та еволюцію даної технології та її можливості до розширення. Однак, що Deception, що Honeypot являються централізованими системами, які все ще мають всі недоліки централізованого підходу, а саме сервер управління. При виявленні основної ланки, хакер може скоригувати свої дії. Нівелювати даний ризик можна побудувавши систему захисту, яка не буде залежати лиш від одного центрального вузла [82]. Blockchain - це багатовузлова система, в якій кожен вузол повинен підтвердити інформацію, яка надходить до однієї з ланок перед тим як впускати її в загальний потік даних. Властивість динамічної зміни й валідування вузлів Blockchain може багатократно підсилити системи захисту та запобігти проблемі централізованого керування. На основі цього необхідно змоделювати динамічну розподілену систему управління, використовуючи динамічні властивості Blockchain та дослідити параметри даної системи. Тому представимо динамічну динамічну розподілену модель прогрманої приманки, утворену N хостами та чотирма службами. Як показано на рис. 3.1. Існує два учасники: хакер і легітимний користувач, який синхронізований з реальним сервісом (тобто клієнт може зберігати місцезнаходження за допомогою реального сервісу і знає точне розташування). N хостів складають приватний Blockchain, який є мережею P2P і не відкриває свої двері для зовнішнього світу.

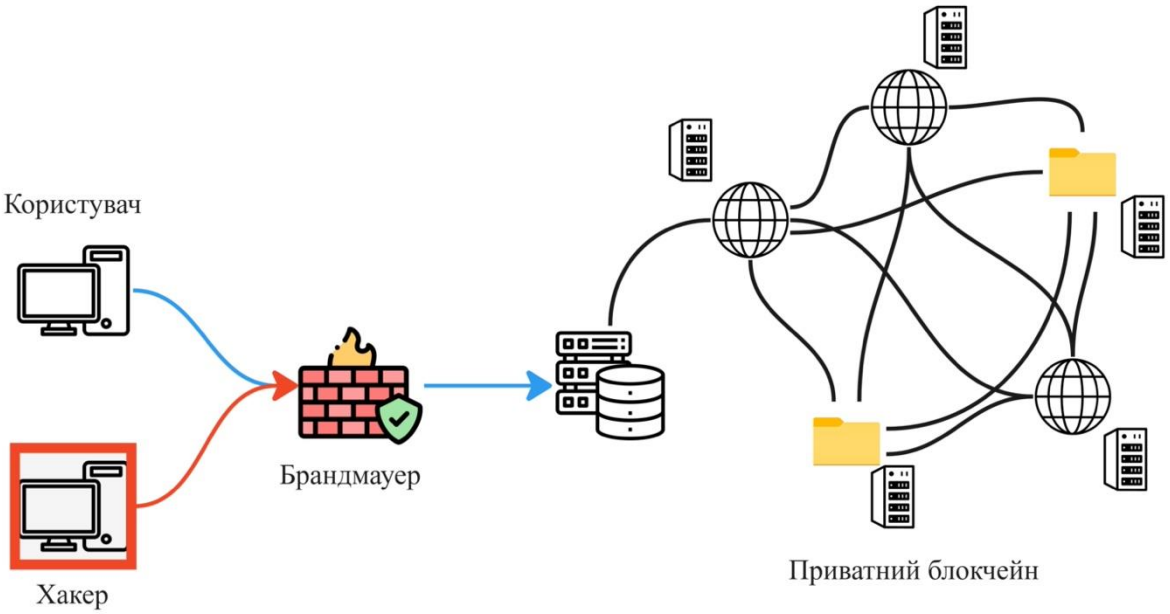

Рис. 3.1. Модель динамічної розподіленої системи програмних приманок

Solana (тобто платформа Blockchain) слугує нижнім шаром в системі. N хостів становлять приватний Blockchain, який утворює мережу P2P. Обчислюючи хеш-значення блоку, хост у приватному ланцюжку може видобути потенційний блок і завантажити його в ланцюг. Цей механізм гарантує розподіл і децентралізацію архітектури розгортання [83]. Тимчасовий головний хост виконує алгоритм розподілу послуг і надсилає відповідну зашифровану інформацію іншим хостам. Як показано на рис. 3.2., у системі майнер блоку $Host_0$ (тобто хост, який успішно обчислює певний хеш) стає основним хостом у період $Table_0$, а інший хост $Host_1$ може замінити $Host_0$ у наступному колі. Хост, який має більші обчислювальні потужності, швидше за все, буде контролером тимчасового центру. Якщо вузько налаштований хост страждає від атак і його продуктивність знижується, він не може служити центральним хостом через брак достатньої обчислювальної потужності, і інші хости замінять його автоматично. Отже, поломка основного хоста $Host_0$ не має значення для всієї системи (тобто система функціонує нормально). Журнали атак, записані одним хостом, завантажуються в Blockchain, а інші вузли синхронізують ці журнали в нашому приватному ланцюжку. Таким чином, кожен вузол має повні дані, які

зберігаються в безпечному та захищеному вигляді для подальшої криміналістики атак.

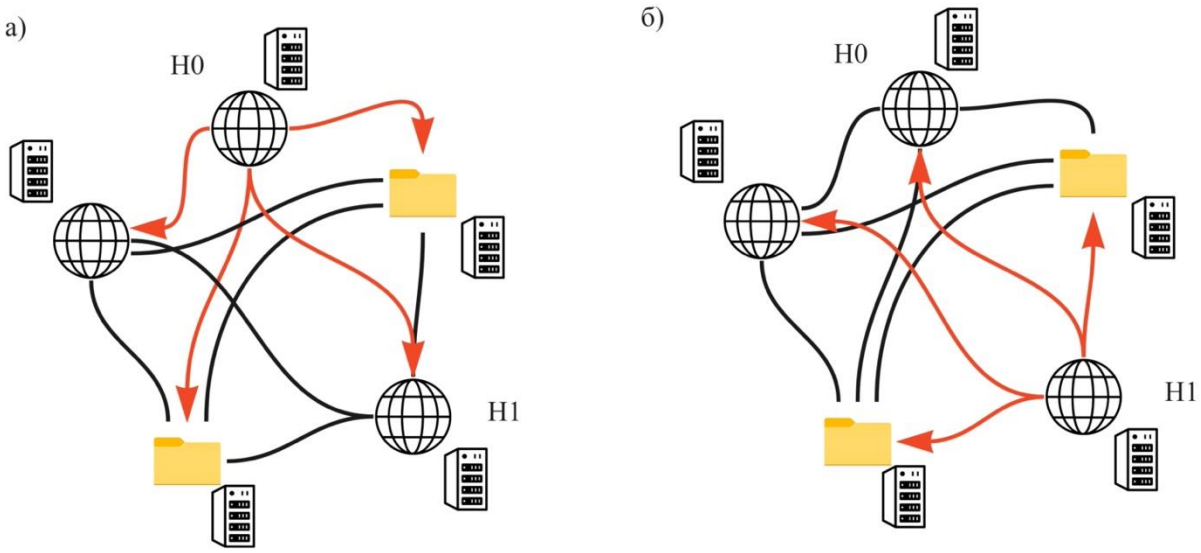

Рис. 3.2. Різні основні хости: 1) Основний хост у $Table_0$, 2) Основний хост у $Table_1$

3.1.1. Опис комунікації хостів в побудованій мережі Blockchain

Хост, який видобуває блок, діє як непостійний контролер центрування. Цей центральний хост генерує інформацію про перетворення, яка призначає кожному хосту для запуску різних служб (тобто для запуску реальної служби або сервісу програмної приманки) відповідно до алгоритму випадкового генерування. Дані містять номери сервісів і кодування 01, які будуть зашифровані за допомогою 2048-бітного алгоритму шифрування RSA [84]. Потім зашифровані дані надсилаються іншим хостам, хостом тимчасового центру в даній приватній мережі. Після прибуття на відповідний хост інформація розшифровується і отримується звичайний текст. Для кодування 01 - нуль є символом запуску сервісу програмної приманки, а одиниця представляє реальний сервіс. За допомогою тексту виконується бітове порівняння, далі запускається вказаний сервіс для завершення процедури виконання. Для авторизованого користувача синхронізація виконується для підтримки нормальної роботи. Відправляючи користувачеві зашифровану інформацію реального сервісу, сервер може надавати звичайний сервіс. Крім того, користувач може надіслати зашифровані дані запиту

«whois + ім’я сервера», щоб активно отримати потрібну адресу певної служби. Таким чином, дійсний користувач може отримати доступ до реальних системних ресурсів під час використання сервісу.

Формальний опис механізму децентралізованого зв'язку на рис. 3.3. виглядає так:

- У певний момент часу тимчасовий основний хост $mHost_j$ запитує про нову базу coinbase до Blockchain через інтерфейс web3J. Coinbase представляє хост, який успішно майнить блок. Після цього $mHost_j$ генерує команду під назвою $Command_{update}$ для її оновлення. $Command_{update}$ має певний формат. $K_{public} = (E, N)$ та $Enc_1 = ((Command_{update})^E mod N)$ обчислюється. Зашифроване повідомлення Enc_1 надсилається кожному іншому хосту в приватному Blockchain. Після отримання повідомлення ці хости з $K_{private} = (D, N)$ обчислять $Dec_1 = ((Enc_1)^D mod N)$. Після перевірки Dec_1 у певному форматі coinbase буде оновлено на кожному хості. Поєднаний з ним хост діє як новий тимчасовий головний хост. Тим часом j в $mHost_j$ змінюється на нове значення.

- Новий основний хост $mHost_j$ має право виконувати алгоритм розподілу. Генеруються номери служб і коди 01, які спрямовують інші хости на відкриття або закриття. Вони вважаються службовими кодами. Відправляється повідомлення $Command_{changeSrv}$, яке містить коди служби. Різні хости отримують різні повідомлення $Command_{changeSrv}$. $Enc_1 = (Command^E_{changeSrv} mod N)$обчислюється і надсилається на $cHost_i$, який представляє загальний хост. $cHost_i$ виконує $Dec_2 = ((Enc_2)^D mod N)$ і отримує просте повідомлення. Dec_2 встановлено, і хост буде відкривати та закривати відповідні служби.

- Хост клієнта надсилає команду запиту на один із цих серверів. Команда $Request_{srv}$ містить повідомлення: ‘хто є Apache’. $Request_{srv}$ шифрується як $enc_1 = ((Request_{srv})^e mod n)$з відкритим ключем $k_{public} = (e, n)$ і пересилається на сервер.

- Сервер розшифровує повідомлення enc_1 через свій закритий ключ $k_{private} = (d, n)$. $dec_1 = ((enc_1)^d mod n)$ виводиться та перевіряється. На сервері є

набір повідомлень запитів $\{R_0, R_1, R_2, R_3\}${R0, R. Якщо $S = dec_1 \oplus R_a = 0$, а ∈ [0, 3], запитувана IP-адреса IP_r в $enc_1 = ((IP_r)^e modn)$ буде повернута клієнту, а його IP-адреса буде додано до основного списку $List_{client} = \{IP_{c1}, IP_{c1}, \ldots IP_{cc}\}$. Інакше значення dec_1 буде проігноровано.

•Після отримання IP_r у $dec_2 = ((enc_2)^d modn)$, клієнтський хост підключиться до цієї IP-адреси для отримання реальних ресурсів.

•Через змінні в різні періоди $\{T_1, T_2, T_3, T_t\}$, справжня IP-адреса буде оновлено до IP_f. Хост, налаштований на IP_r, надсилає клієнтам команду $Update_{src}$ відповідно до $List_{client}$. Оновлений IP_f шифрується як $enc_3 = ((Update_{src})^e modn)$.

•Обчислюється $dec_3 = ((enc_3)^d modn)$. Існують чотири команди $\{C_0, C_1, C_2, C_3\}$ які слідують спеціальному формату в клієнтах. Якщо $s = dec_3 \oplus C_a = 0$, а ∈ [0, 3], клієнт підключається до нового IP_f. Періодично перемикаючи сервіси, буде виконуватися циклове виконання згаданих кроків.

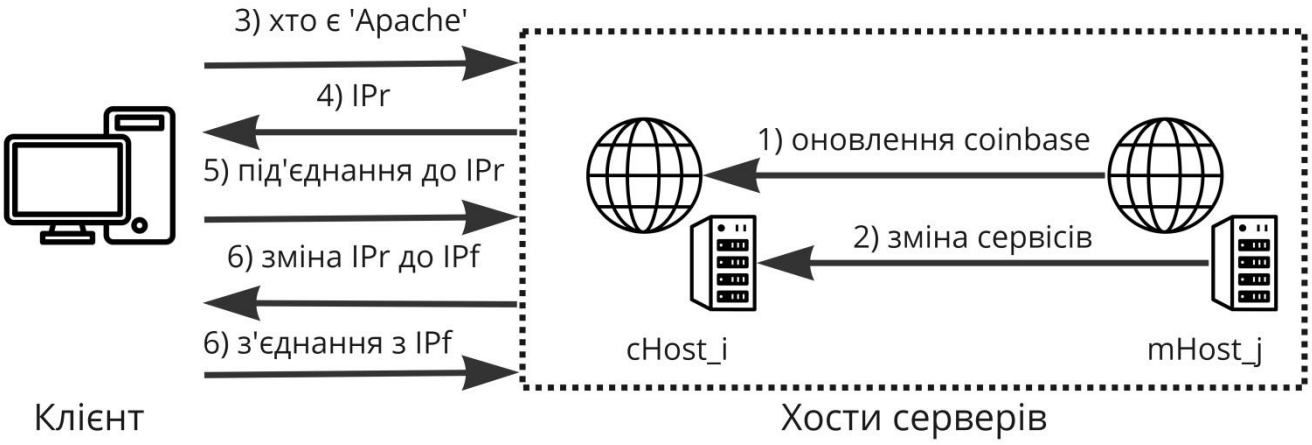

Рис. 3.3. Децентралізований механізм комунікації

3.1.2. Трансформація сервісів під час під'єднання до вузлів

У даній схемі існує всього чотири види сервісів, і кожен сервіс має як справжні, так і підроблені атрибути (тобто чотири реальні сервіси та чотири відповідні (підроблені) сервіси). Періодичне перемикання сервісів виконується кожен $Table$ період. Порівняння розподілу сервісів показано на рис. 3.4. Обидва види сервісів постійно змінюються [85]. Виходячи з передумови існування

технології ідентифікації anti-honeypot, існує три види застосувань в системі захисту:

- Якщо сервіс $Service_0$ на хості $Host_0$ реальна в $Table_0$, $Service_0$може стати підробленим в наступному періоді $Table_1$. Після трансформації зловмисник не може отримати доступ до реальних ресурсів служби в $Table_1$.
- Якщо мервіс $Service_1$ на хості 1 служить приймаючим сервісом в $Table_0$, згідно з обіцянкою технології anti-honeypot, як тільки зловмисник виявить, що сервіс є пасткою, він уникне $Service_1$, який може змінитися на реальний сервіс в $Table_1$. Таким чином, це перешкоджає зловмиснику отримати доступ до реальних ресурсів.
- Якщо служба $Service_0$ на хості $Host_0$ реальна в $Table_1$, через синхронізацію з дійсними користувачами, клієнт надсилатиме запити лише до реальної служби. Оскільки існують деякі підроблені служби (наприклад, honeypots), будь-який трафік доступу до honeypots $Service_0$, $Service_1$, . . . , $Service_n$ в $Host_0$, $Host_1$, . . . , $Host_m$ позначено як запис атаки.

Таким чином, трансформація та переміщення сервісів та служб спантеличує зловмисників і захищає розроблену систему.

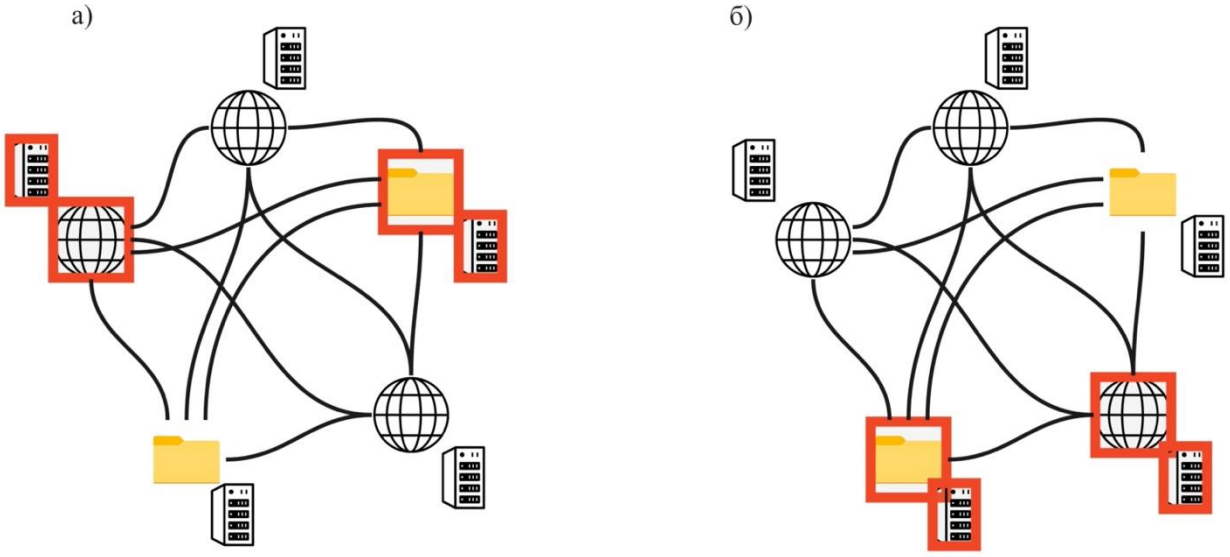

Рис. 3.4. Порівняння: 1) Основний хост у Table_0 перед переміщенням, 2) Основний хост у Table_1 після переміщення

3.2. Розробка методу динамічної системи побудованої за допомогою програмних приманок

У цьому розділі представлено метод динамічної системи побудованої за допомогою програмних приманок [86]. В основі цього методу виділено модель та математичний опис який представлено на рис. 3.5 та наступними кроками:

1. Коли запити проходять через брандмауер, програмне забезпечення в маршрутизаторі класифікує шкідливі запити та активує процес розгортання програмних приманок.

2. Шкідливі запити будуть автоматично перенаправлені на щойно розгорнуту приманку.

3. Приватний Blockchain Solana створено для обміну деталями розгортання всіх сервісів і приманок у піднятій мережі.

4. Піднімається система, яка динамічно розгортає необхідну службу на кожному вузлі абсолютно випадковим чином. Програма розподілу служб, що працює на тимчасовому головному сервері, дбає про цю функціональність, взявши адресу облікового запису, що використовується кожним сервером у ланцюзі блоків для доступу до нього, як вхідні дані, і призначивши код служби, який є серією двійкових кодів - цифр, що дорівнює номеру служби, включаючи індивідуальні служби приманки. Приклад: беруться до уваги чотири служби, такі як Apache, MySQL і відповідні служби програмної приманки, тоді код служби: 0000, де 0 означає ввімкнено, а 1 означає вимкнення.

5. Програма розподілу генерує деталі розгортання. Вони мають бути безпечно спільно використані між серверами тимчасових клієнтів. Тому використовується Blockchain як спільне сховище для цієї мети.

6. Кожен із серверів представляє вузол приватного Blockchain. Blockchain синхронізує дані в усіх вузлах, тому дані, збережені з одного вузла, будуть доступні для інших.

7. Щоб зберігати та отримувати дані з Blockchain, описана функція смарт-контрактів, що є розгорнута в приватній мережі Blockchain. Їх можна

використовувати для збереження та отримання деталей розгортання, коли це необхідно.

8. З постійними інтервалами один із тимчасових клієнтів стає тимчасовим сервером, а попередній ведучий стає клієнтом. Новий тимчасовий сервер знову перерозподілить усі служби випадковим чином.

Програма розподілу послуг і смарт-контракти працюють на різних платформах. Тому використовується інтерфейс, наданий Web3js, щоб встановити з'єднання між ними. Основний хост знаходиться на першому рівні Blockchain в першому вузлі, який в той же час є тимчасовим через властивість зміни позиції.

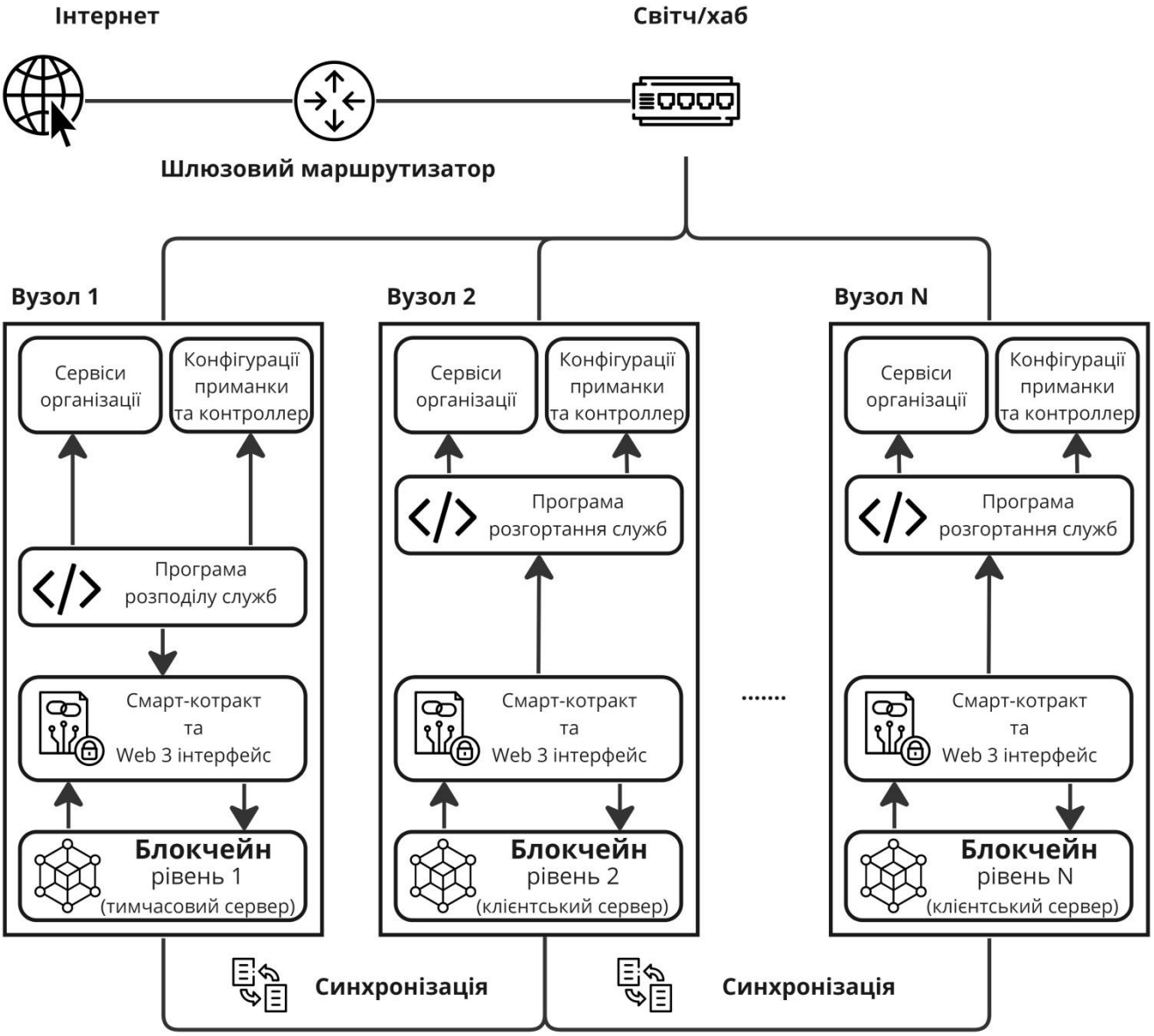

Рис. 3.5. Метод динамічної системи

побудованої за допомогою програмних приманок

Як формальна мова специфікації, Alloy можна використовувати для вираження системи на основі логіки першого шару [87]. Створення огляду системи, теорії встановлення та абстрагування атомів і співвідношень реалізовано за моделлю системи Alloy.

Відповідно, введено поведінку системи та деякі пов'язані позначення зображені в таблиці 3.1.

Система виконає набір атомарних дій, тобто дії = {генерувати, надсилати, отримувати, чекати, відкривати, закривати, відновлювати, скомпрометувати}. Ці дії та їх параметри зведені в Таблицю 3.2.

Таблиця 3.1.

Позначення які використовуються для моделювання системи.

Назва	Позначення
Стани	$\{s_n, s_c, s_b\}$
Канали	$\{c_1, c_2, \ldots, c_c\}$
Хости	$\{h_1, h_2, \ldots, h_h\}$
Порти	$\{p_1, p_2, \ldots, p_p\}$
Ідентифікатори	$\{id_1, id_2, \ldots, id_h\}$
Служби	$\{{srv_1}^R, {srv_1}^H, \ldots, {srv_s}^R, {srv_s}^H\}$

Таблиця 3.2.

Опис дій системи.

Функція	Опис
generate(data)	хост генерує дані
open(srv_i)	хост відкриває службу

close(srv_i)	хост закриває службу
send (data,c_i)	хост відправляє дані через канал
receive (data,c_i)	хост отримує дані через канал
compromise()	хост скомпроментовано
recover()	хост відновлено
wait ($reply_i$)	хост очікує на відповідь після відправлення $reply_i$

Служби на хості абстрагуються подіями введення/виводу. Події generate(data) та send (data,c_i) представляють вихідні дані сервісів, а подія receive (data,c_i) являє собою отримання даних. З точки зору безпеки, кожен хост може працювати в нормальному або скомпроментованому режимі одночасно. Нормальний режим означає, що хост працює без шкідливих даних і підтримує звичайну роботу. Однак скомпроментований режим вказує на те, що хост працює зловмисним чином і завдає собі шкоди. Стани, що стосуються запущеного $host_i$, поділяються на дві категорії: $Service_i^N$ для звичайного режиму та $Service_i^C$ для скомпроментованого режиму [88]. Найгірша ситуація полягає в тому, що хост зламався і перестав працювати в режимі поломки $Service_i^B$. Таким чином, стани складаються з трьох типів $\{service_n, service_c, service_b\}$. Хост працюватиме як поточний режим, поки не відбудеться перехід → InterT, який представляє відносини переходу між трьома різними режимами. Переходи, показані на рис. 3.6, можна визначити так:

$$Service_a \to T, InterTService_b : Service_a \in Service^N \wedge Service_b \in Service^C$$

$$Service_b \to T, InterTService_a : Service_b \in Service^C \wedge Service_a \in Service^N$$

$$Service_b \to T, InterTService_c : Service_b \in Service^C \wedge Service_c \in Service^B$$

$$Service_c \to T, InterTService_a : Service_c \in Service^B \wedge Service_a \in Service^N$$

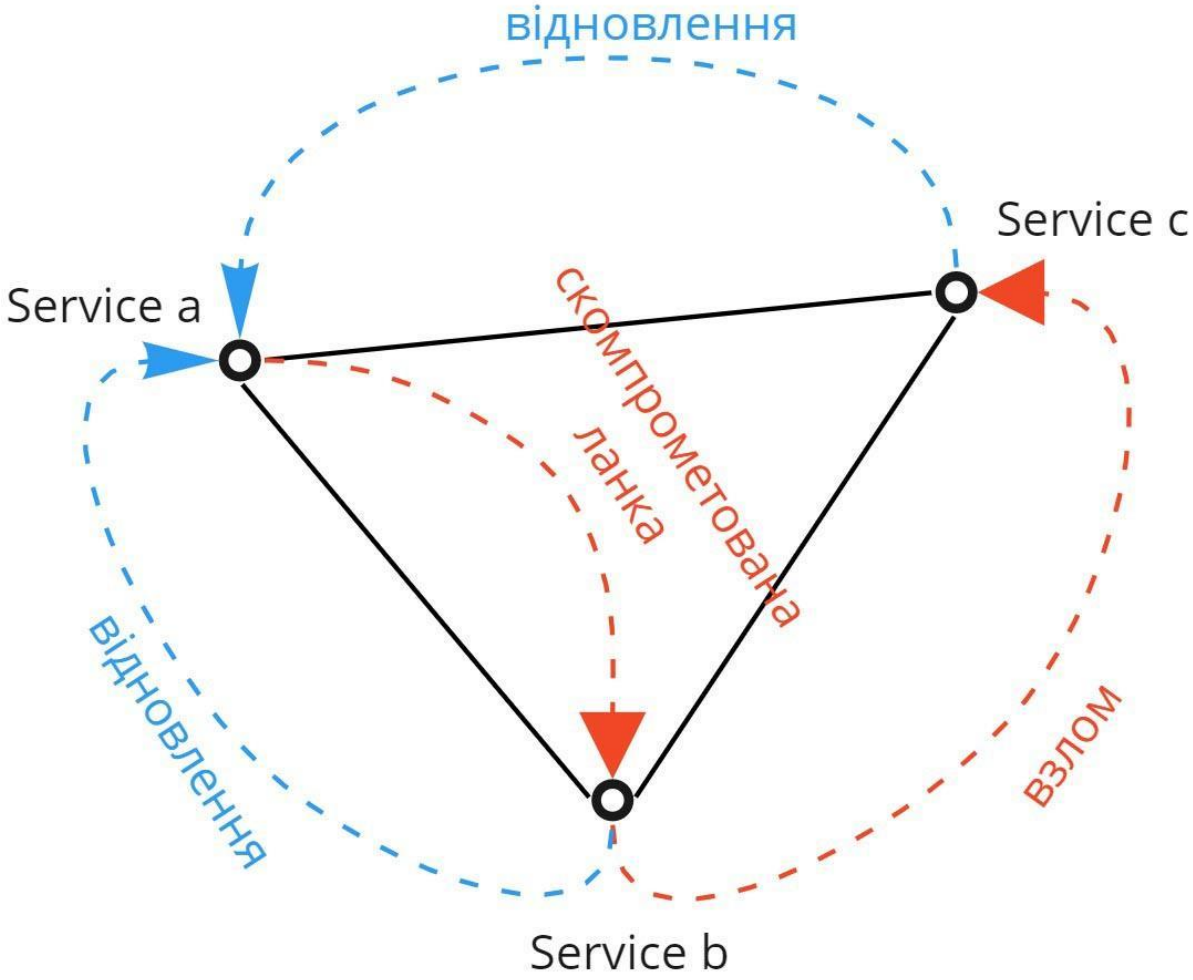

Рис. 3.6. Перехідні стани ланок

Хост складається з п'яти частин $host_i = (id_i, Ports_i, Services_i, States_i, \rightarrow T)$, де id_i — ідентифікатор хоста, $Ports_i$ — набір портів, $Services_i$ — набір служб, $States_i$ — набір станів і → T — це набір відношень переходу, наведений у таблиці 3.3.

В основі комунікації даних технології Blockchain лежить механізм консенсусу. Ланки-валідатори перевіряють дані, що надходять з інших ланок та шляхом голосування вирішують чи допускати інформацію в мережу. Будь-яке відхилення (неверифікація даних) компрометує ланку з якої надходить запит. Тому навіть атаки внутрішнього характеру (якщо зловмисник в мережі) дуже складно провести непоміченими.

Таблиця 3.3.

Перехідні залежності станів ланок.

Перехід	Позначення
$\rightarrow Table$	$\rightarrow IntraT \cup \rightarrow InterT$
$\rightarrow IntraT$	{звичайний стан - звичайний стан} {атакований стан - атакований стан} {скомпрометований - скомпрометований}
$\rightarrow InterT$	{атакований стан - звичайний стан} {скомпрометований - звичайний стан} {скомпрометований - звичайний стан} {скомпрометований - атакований стан}

Оскільки дані, що передаються між хостами, гарантують нормальну роботу системи, вони відіграють важливу роль в аналізі безпеки. Передбачається, що кожна частина даних генерується лише одним хостом [89]. Дані можуть бути шкідливими і містять деякі команди, які призводять до шкідливої діяльності. Хост, який створює шкідливі дані, вважається зламаним. Вони описуються таким чином:

$$malicious(data) = \exists s^{generate(data)} \rightarrow Tables' : service \in Service^C$$

$$compromised(h_i) = \forall s \rightarrow Table, IntraTs' : service \in Service^C \wedge service' \in Service^C$$

Хост h_i із нормальною поведінкою перебуває в нормальному режимі $normal(hi) = compromised(h_i)$. Припускається, що якщо звичайний хост отримує шкідливі дані, він перейде в режим компрометації, ще більше компрометуючи себе. Щоб змоделювати поширення неавторизованих даних під час атаки, виходить наступне:

$$service_i^{compromise} \rightarrow Table, InterTService_i \exists Service_{i-1} : Service_{i-1} \rightarrow Table, IntraTservice_i$$

Щоб скомпрометований хост не перехопив дані, що передаються під час зв'язку, канал зв'язку має бути захищений:

$$secure(c) = \forall(h_j, h_i) connectedState(h_j, h_i, c, States) \wedge \neg\exists s^{accept(data)}$$
$$\rightarrow InterTs' \wedge (compromised(h_i) \vee compromised(h_j))$$

Система складається з h хостів, як зазначено в таблиці 5. Усі стани в цих хостах ілюструють загальний стан системи, тобто $Service_{system} = Service_{h1} \cup Service_{h2} \cup \ldots \cup Service_{hh}$. Безперервна авторизована поведінка кожного хоста (наприклад, надсилання даних через канал) забезпечує нормальне функціонування системи. Будь-які хости спілкуються один з одним шляхом надсилання та отримання даних через канали зв'язку:

$$host_i, send(c, data) \rightarrow host_j, receive(c, data)$$hhi,send(c, data).

Це вказує на те, що спільне використання одного каналу дозволяє як підключатися, так і передавати спільні дані. Обмін даними може здійснюватися лише тоді, коли вони з'єднані через один канал зв'язку. Отже, ми визначаємо таке твердження:

$$connected(host_i, host_j, c, States) = \exists host_j^{connect(c)} \rightarrow host_i \wedge host_i^{connect(c)}$$
$$\rightarrow host_j$$

Під час процесу комунікації поведінка $host_i$ та $host_j$ показана на рис. 3.7.

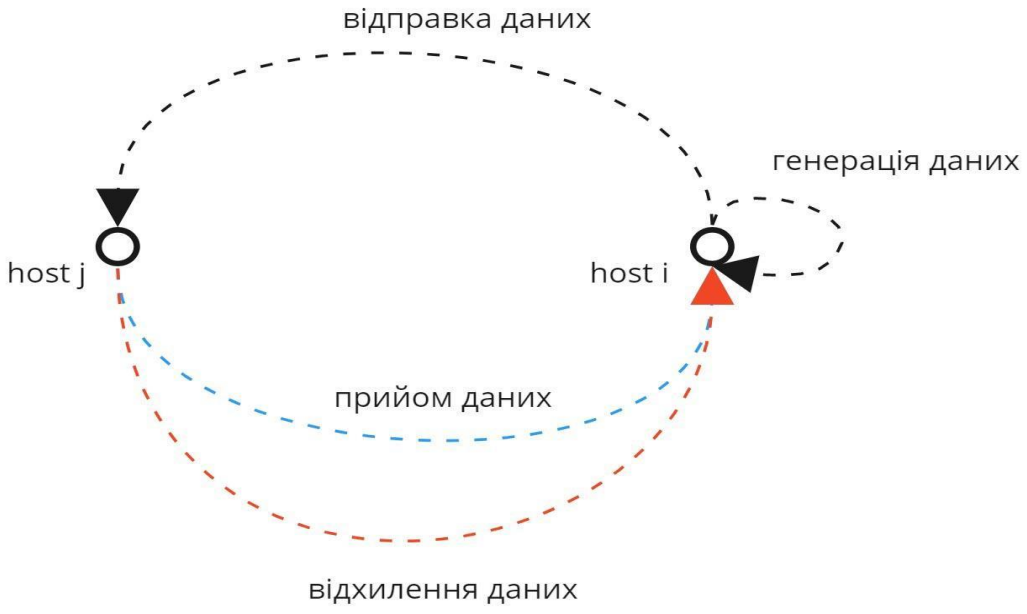

Рис. 3.7. Поведінка під час комунікації

Хост $host_i$ генерує дані та надсилає дані до $host_j$ через канал зв'язку. Після отримання даних від $host_i$, $host_j$ може вирішити, прийняти чи відхилити ці дані. Як тільки $host_j$ отримує та приймає шкідливі дані, він стає скомпрометованим:

$$host_i, generate(data)$$

$$host_i, send(c, data) \rightarrow host_j, receive(c, data)$$

$$host_j, accept(data) \vee host_j, discard(data)$$

Якщо: $malicious(data) \wedge accept(data)$

Тоді: $host_j, compromise$

Отже: $compromisedState(h_j)$

Звичайний хост служить легальною частиною системи і забезпечує нормальне функціонування його служб для користувачів. Як згадувалося вище, {скомпрометований - звичайний стан} вказує, що скомпрометований хост стає звичайним під час дії відновлення:

Якщо: $compromisedState(host_j)$

Тоді: $host_j, recover$

Отже: $normalState(host_j)$

Система абстрагується та зосереджується на передачі даних для наступного аналізу атак щодо питань безпеки [90].

Висновки до розділу 3

Таким чином, у третьому розділі дисертаційної роботи було розроблено та запропоновано метод використання програмних приманок побудованих на основі технології Blockchain. Проведене дослідження дозволяє виділити наступне:

1. Представлений метод динамічної системи заснований на використанні програмних приманок та приватного Blockchain Solana для захисту від шкідливих запитів та представлений функцією динамічного розгортання необхідних служб на кожному вузлі. Програма розподілу служб та смарт-контракти забезпечують безпечну спільну роботу між серверами тимчасових клієнтів, а інтерфейс Web3js

дозволяє забезпечити зв'язок між різними платформами. Цей метод може забезпечити надійний та безпечний захист мережі від шкідливих впливів та ефективної роботи служб на кожному вузлі.

2. Розроблена модель децентралізованого зв’язку вузлів за рахунок вдосконаленого математичного апарату допомагає оптимізувати процеси класифікації шкідливих запитів та розподілу служб на вузлах мережі взявши адресу облікового запису, що використовується кожним сервером у ланцюзі блоків для доступу до нього, як вхідні дані, і призначивши код служби, який є серією двійкових кодів. Клієнт створює два комунікаційні потоки та відкриває відповідні порти, таким чином виступаючи як сервер. Клієнт може отримувати дані сервера і надсилати запити на сервер. Існує два типи шаблонів для отримання реальної службової інформації: позитивний шаблон і пасивний шаблон. Для позитивного шаблону клієнт активно надсилає запити «whois + ім’я сервера» на сервер для певної IP-адреси служби. Однак у пасивному шаблоні клієнт чекає лише даних від сервера, який повинен прочитати список клієнтів і надіслати їм правильний IP усіх служб. Крім того, є два ключові файли (publicKey і privateKey), які містять 2048-бітні дані в клієнті, шифровані RSA. Файл publicKey використовується для шифрування даних перед процесом зв'язку, а файл privateKey може дешифрувати зашифровані дані, отримані від сервера. Отримавши реальну інформацію про сервіс, клієнт автоматично підключиться до потрібного IP і отримає реальні ресурси.

РОЗДІЛ 4. ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДУ ВИКОРИСТАННЯ ПРОГРАМНИХ ПРИМАНОК ПОБУДОВАНИХ НА ОСНОВІ ТЕХНОЛОГІЇ BLOCKCHAIN ЯК ЕЛЕМЕНТІВ ЗАХИСТУ

4.1. Аналіз безпекового рівня системи використання програмних приманок на основі розробленої моделі та рішення щодо його захисту

За останні кілька років було багато проблем із безпекою. Атаки представляють дії із застосуванням насильства або чогось незаконного, щоб спробувати пошкодити систему безпеки мережі. Технології захисту від усіх видів атак забезпечують стабільність системи. У цьому розділі в основному аналізується три види атак (тобто атаки сніферів, атаки сканування та DDoS-атаки) та відповідні рішення для вирішення цих проблем [91].

По-перше, ці атаки є досить поширеними в сучасних мережах і ними користуються зловмисники для здійснення різних видів кіберзлочинності. Атаки сніферів, атаки сканування та DDoS-атаки можуть бути використані для здійснення різних видів кіберзлочинності, таких як крадіжка конфіденційної інформації, пошкодження веб-сайтів, вимагання виплати викупу та багато іншого.

По-друге, ці атаки відрізняються своєю специфікою та характеристиками. Наприклад, атаки сніферів передбачають перехоплювання трафіку, що може призвести до розкриття конфіденційної інформації, а атаки сканування полягають у пошуку вразливостей у мережі та їх використанні зловмисниками для здійснення атак, а DDoS-атаки полягають у спробі зупинити роботу веб-сайту шляхом перевантаження його серверів [92].

По-третє, використання цих атак дозволяє випробувати різні технології захисту та оцінити їх ефективність. Наприклад, ефективність технологій захисту від атак сніферів може бути оцінена шляхом випробування методів шифрування даних, а ефективність технологій захисту від DDoS-атак може бути оцінена шляхом випробування методів розподіленої обробки запитів.

4.1.1. Опис проведення експерименту: Сніфер атака

Сніфер атака – це незаконна дія третьої сторони з метою таємного отримання даних, які передаються між обома сторонами зв'язку. Щоб продемонструвати такий тип атаки, береться до уваги два хости $host_0$ та $host_1$. Обидва підключені через один канал зв'язку $connectedState(host_0, host_1, c, States)$та служать звичайними хостами $normalState(host_0) \wedge normal(host_1)$. У сценарії атаки з використанням сніферу існує третя атака хоста, і відбувається перехоплення інформації. Коли $host_0$ надсилає дані на $host_1$, $host_{attack}$ таємно отримує передані дані (тобто, $connectedState(host_0, host_{attack}, c, States)$), які мають бути відправлені єдиному одержувачу $host_1$. Такий напад можна описати так:

$$[host_0]generate(data)$$

$$[host_0]send(c, data)$$

$$[host_1]receive(c, data)$$

$$[host_{attack}]receive(c, data)$$

Неминуче, що комунікаційні дані можуть бути отримані нелегальним зловмисником. Ми повинні намагатися не допустити розкриття інформації. Тому для гарантії безпеки каналу зв'язку слід виконувати шифрування даних. Щоб захистити від атаки сніферів між $host_0$ та $host_1$, потрібен захищений c для даних, тобто $connecte(host_0, host_1, c, States)$ має бути захищений:

$$\forall: secure(c)$$

Розроблена система зашифрована RSA 2048 - бітовим алгоритмом шифрування, і його неможливо декодувати без відповідного ключа конфіденційності. Таким чином, канал зв'язку захищений у запропонованій системі і додатково запобігає атаці сніферів.

4.1.2. Опис проведення експерименту: атака сканування

Атака сканування являє собою дії під час яких відправляються запити на всі порти цільового хоста з метою дослідження відкритих портів і подальшого використання їх помилки для запуску атак. Зловмисники зазвичай

використовують інструмент сканування для здійснення атаки сканування. Щоб продемонструвати атаку сканування, припускається, що хост $host_2$ сканується за допомогою $host_{attack}$. Перш за все, атака надсилає запит даних на всі можливі порти на цільовому хості $host_2$. По-друге, відкриті порти отримають запит даних і відповість на вихідний хост $host_{attack}$. Далі $host_{attack}$ отримує ці відповіді. Нарешті, $host_{attack}$ знаходить помилки $host_2$ і запускає атаки відповідно до відкритих портів. Такий напад можна описати так:

$$[host_{attack}]generate(request)$$

$$[host_{attack}]send(c, request)$$

$$[host_2][port_i]receive(c, request)$$

$$[host_2][port_i]generate(reply)$$

$$[host_{attack}]receive(c, reply)$$

$$[host_{attack}]compromiseState(host_2)$$

Щоб усунути проблеми, викликані атакою сканування, скановані порти повинні бути непередбачуваними для зловмисників. Щоб підтримувати властивість непередбачуваності, ці порти постійно змінюються. Якщо зловмисник сканує відкритий порт $port_i$ та має намір атакувати хост $host_2$ за допомогою$normalState(host2)$, захисник повинен закрити $port_i$ та відкрити ще один $port_j$, що вказує на те, що відсканована інформація про порти втрачає свою цінність. Насправді, служба в $host_2$ має власний номер порту:

$$host_2 open(srv_i) = host_2, open(port_i)$$

$$host_2 close(srv_i) = host_2, close(port_i)$$

Отже, щоб захистити служби від атаки сканування, усі порти потрібно періодично закривати та відкривати:

$$\forall p: close(port_i) \vee open(port_j)$$

У розробленій системі є чотири служби, які періодично відкриваються або закриваються їх на різних хостах для захисту від атак сканування.

4.1.3. Опис проведення експерименту: DDoS атака

DDoS-атака являє собою модель атаки для відправки великої кількості запитів цільовому хосту. Хост отримує тимчасовий сплеск запитів, і відбудеться поломка. Нелегальна атака на хост генерує багато запитів і надсилає їх хосту $host_3$ за допомогою $normalState(host3)$. Отримавши ці запити від $host_{attack}$, $host_3$ буде чекати їх відповідей. Від них не буде жодної відповіді, що свідчить про марнотратство системних ресурсів і подальше споживання ресурсів $host_3$, поки він не буде розбитий. Такий напад можна описати так:

$$[host_{attack}]generate(request1)$$
$$[host_{attack}]generate(request2)$$
$$\dots$$
$$[host_{attack}]generate(requestn)$$
$$[host_{attack}]send(c, request1)$$
$$[host_{attack}]send(c, request2)$$
$$\dots$$
$$[host_{attack}]send(c, requestn)$$
$$[host_3]receive(c, request1)$$
$$[host_3]receive(c, request2)$$
$$\dots$$
$$[host_3]receive(c, requestn)$$
$$[host_3]send(c, reply1)$$
$$[host_3]wait(reply1)$$
$$[host_3]send(c, reply2)$$
$$[host_3]wait(reply2)$$
$$\dots$$
$$[host_3]send(c, replyn)$$
$$[host_3]wait(replyn)$$
$$[host_{attack}]breakdown(host_3)$$

Вихід з ладу $host_3$ призводить до єдиної точки відмови. Для вирішення проблеми слід взяти до уваги розподілену схему. У порівнянні з традиційним

централізованим хостом, розподілена система може впоратися з проблемою єдиної точки відмови. Розподілена система містить h хостів та $host \geq 2$. Коли $host_{attack}$ надсилає n запитів, існують дві можливі ситуації для розподіленої системи:

- DDoS-атака на один хост. У такому випадку відбувається поломка хоста $host_3 : breakdown(host_3)$. Навіть якщо $host_3$ не може функціонувати, інші хости (тобто $host_1$, $host_2$,$host_4$,...,$host_h$ з $normalState(host1) \wedge normalState(host_2) \wedge normalState(host_4) \wedge normalState(host_h)$ все ще можуть забезпечити обслуговування користувачів і підтримання нормального функціонування всієї системи, що дозволяє уникнути однієї точки збою.
- DDoS-атака на всі хости. У такому випадку приймається n як максимальну кількість запитів на аварійне завершення роботи хоста, і кожен хост ділиться трафіком атаки. Якщо $host_{attack}$ надсилає n запитів, кожен хост отримує запити n/h. h розподілені хости значно зменшують незаконний потік в порівнянні з одним хостом, що вказує на те, що хости в системі не зазнають збою.

Якщо традиційна система стикається з DDoS-атакою, то:

$$host_{attack}^{DDoS^n} \rightarrow h$$

Якщо розподілена система стикається з DDoS-атакою, то:

$$host_{attack}^{DDoS^{n/h}} \rightarrow h_1$$

$$host_{attack}^{DDoS^{n/h}} \rightarrow h_2$$

$$host_{attack}^{DDoS^{n/h}} \rightarrow h_3$$

$$\dots$$

$$host_{attack}^{DDoS^{n/h}} \rightarrow h_h$$

У прототипній системі є п’ять розподілених і децентралізованих хостів для ефективного пом’якшення DDoS-атаки. Підсумовуючи, було проаналізовано три види атак і проілюстровано рішення для боротьби з цими атаками. Теоретично доведено, що дана схема може захиститися від цих атак.

4.2. Експерементальний аналіз рівня безпеки під час симуляції рішення

Тепер необхідно провести оцінку ефективність розробленої динамічної розподіленої схеми захисту програмних приманок [93]. Реалізація системи-прототипу здійснюється на Python, Java та Solidity (тобто на мові програмування Blockchain), інтерфейс програми контролера зображений на рис. 4.1. Крім того, експерименти проводяться на п’яти персональних комп’ютерах (ПК) Windows 16 ГБ на якій стоїть WM та симулюється операційна система (ОС) Linux з 8 Гб оперативної пам’яті для запуску служб, одному ПК з Windows 32 ГБ на якій стоїть WM та симулюється ОС Linux з 16 Гб оперативної пам’яті для запуску атаки різного масштабу та одному ПК з Windows з 32 Гб оперативної пам’яті для авторизованого користувача. Сервіси (MySQL v8.0.27, Apache v2.4.51, Vsftpd v3.0.5 і Nginx v1.24.4) і Solana v1.6.7 (тобто платформа Blockchain, що використовується для формування приватного Blockchain) встановлені на п’яти серверних хостах. Загальна кількість реальних послуг на різних хостах розраховується для ілюстрації їх середніх розподілів. Проводяться три види тестів атак: сніфер, атака сканування та атака DDoS.

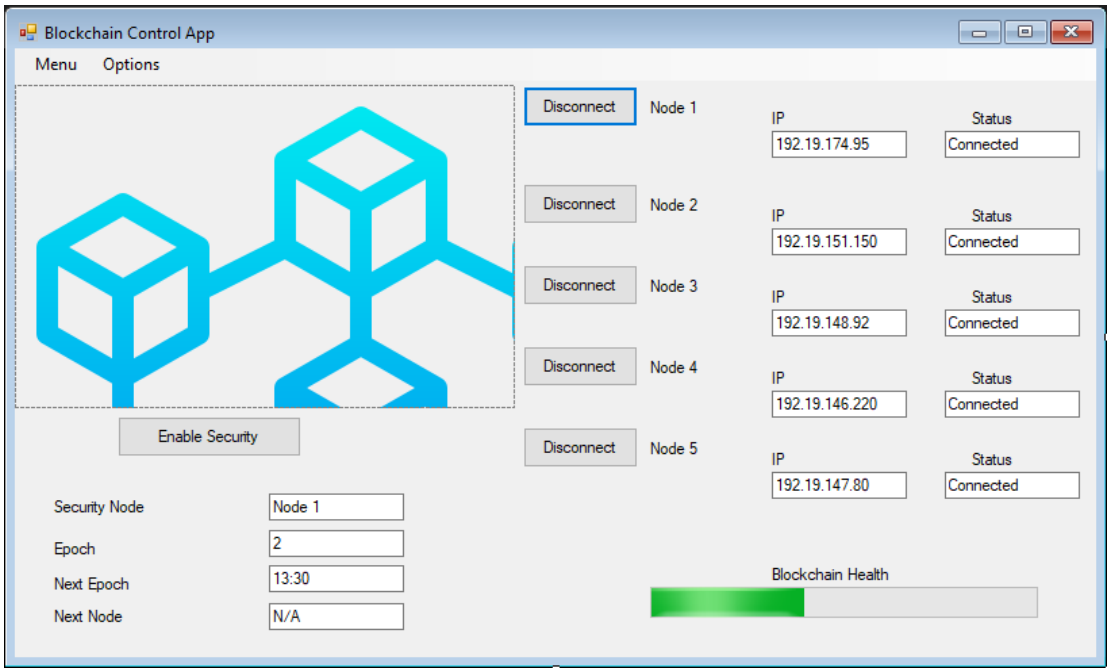

Рис. 4.1. Інтерфейс розробленої програми-контролера системи Blockchain

Через трансформацію сервісів зловмисники не мають уявлення про місцезнаходження справжнього сервісу. Тим не менш, вони можуть спробувати знайти хост із великою часткою реальних сервісів. Таким чином, алгоритм розподілу виконується 60000 разів для тестування розподілу реальних сервісів. Оскільки існує чотири види сервісів, у даному експерименті буде 240 000 реальних сервісів. Точна кількість чотирьох сервісів на п'яти хостах наведена на рис. 4.2. Частка різних хостів по черзі становить приблизно 22%, 20%, 23%, 19% і 16%. Загалом розподіл відбувається з однаковою ймовірністю. Усі ці відсоткові значення становлять понад 16%, що вказує на те, що реальні служби, ймовірно, працюватимуть на кожному хості.

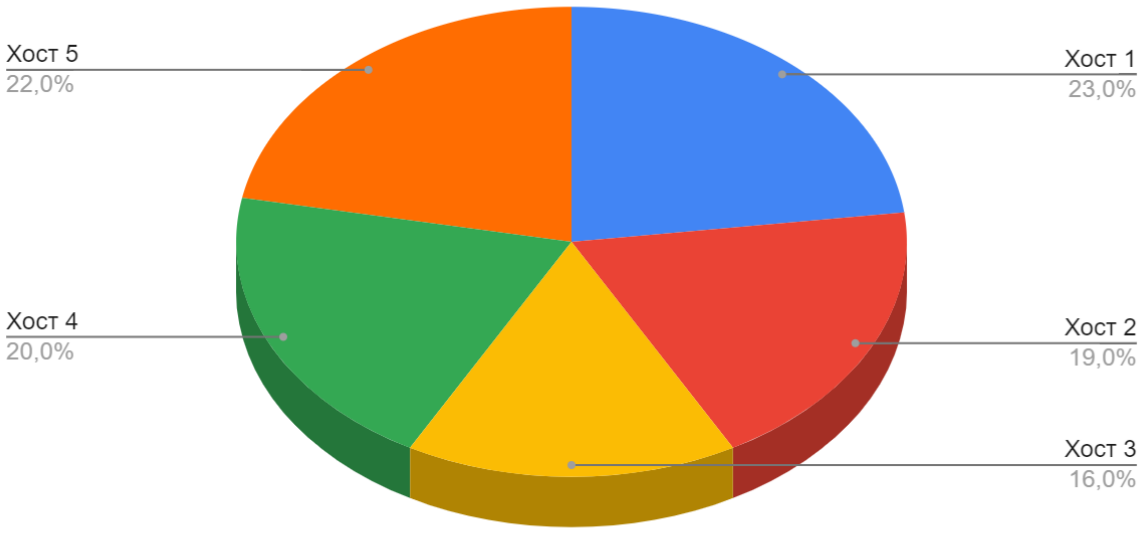

Рис. 4.2. Розподіл реальних сервісів між хостами

4.2.1. Дослідження захисту системи використання програмних приманок на основі розробленої моделі від атаки сніфером

Атака сніфером здійснюється через Wireshark v3.6.0. Як показано на рис. 4.3, дані зв'язку між двома серверами отримують таємно. Як вже згадувалось вище, комунікаційна інформація шифрується за допомогою 2048-бітового алгоритму шифрування RSA. Сніфер атака відбувається не на реальний сервіс, а

на підроблений. Атаку приймає на себе активована програмна приманка. Навіть якщо зловмисник отримує дані, передані під час комунікації, він не може отримати відкритий текст із зашифрованого тексту. Однак навіть якщо припускати, що зловмисник все-таки якимусь чином зміг декодувати дані, то він отримає не справжню інформацію а підроблену, підсунуту йому системою захисту приманки. В такому випадку зловмисник додатково повинен буде витратити колосальну кількість часу для декудоування, що ілюструє, що дана схема ефективна для захисту від атаки сніферу.

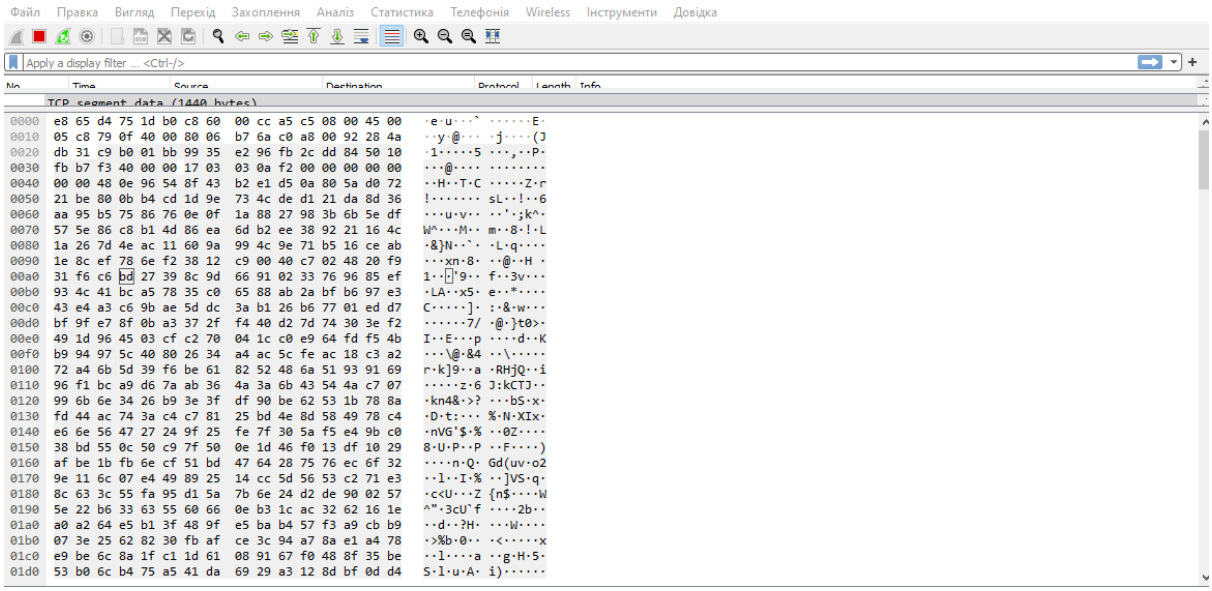

Рис. 4.3. Інформація отримана від атаки сніфером

4.2.2. Дослідження захисту системи використання програмних приманок на основі розробленої моделі від атаки скануванням

Сканування є ефективним кроком для зловмисників для пошуку системної помилки [94]. Зазвичай зловмисник може отримати IP-адресу цілі заздалегідь. У даному експерименті проводиться тестування атаки сканування через Namp v7.92 (тобто інструмент сканування). Перевіряється п'ять згаданих вище хостів-серверів, що показано на рис. 27. Використовується команда сканування nmap -T4 -A -v 192.19.174.95 192.19.151.150 192.19.148.92 192.19.146.220 цих хостів до 192.19.146.220 і навпаки. Взявши, наприклад, IP «192.19.174.95», як показано на рис. 4.4, в результаті виходить, що порти 80 і 21 відкриті. Вони представляють Nginx і Vsftpd відповідно. У цей момент зловмисник може запустити злом

сервера. Однак ми отримуємо результат, що порти закриваються, що показано на рис. 4.5. Оскільки виявлені служби закриті, зловмисник не може отримати жодних сервісних ресурсів. Навіть якщо відкритий той самий порт, зловмисник, швидше за все, отримує доступ до програмної приманки та отримає підроблені ресурси. Завдяки властивості трансформації сервіси постійно змінюються. Навіть використовуючи інструмент сканування, зловмисник не має уявлення про точну IP-адресу служби та потрапляє у пастку програмної приманки. Після цього програмна приманка записує в журнал атак подію та дозволяє фахівцям з кібербезпеки оцінити ризики та довідатися про дії зловмисника. Легітимний користувач не має потреби сканувати порти. Сам факт сканування - це порушення безпеки системи. Тому дана схема ефективна для захисту від атаки сканування, що дає перевагу в комплексному захисті комп'ютерних мереж побудованих використовуючи метод динамічних програмних приманок [95].

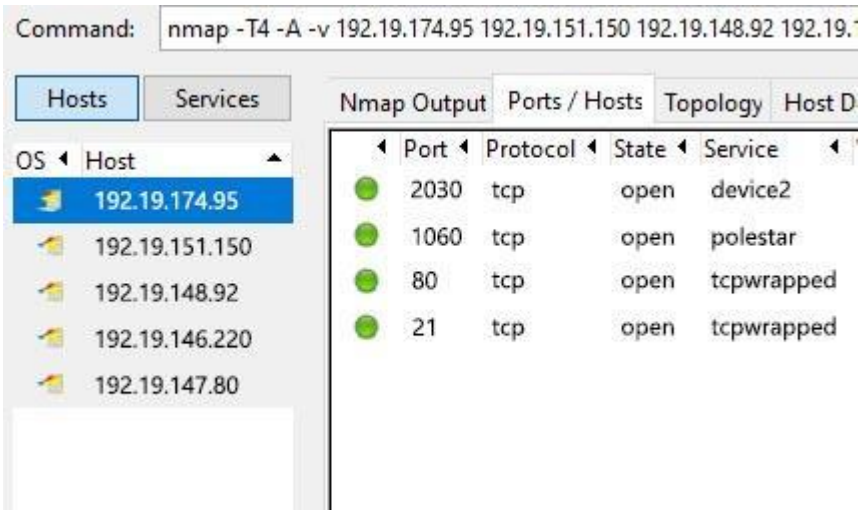

Рис. 4.4. Перше сканування: результат

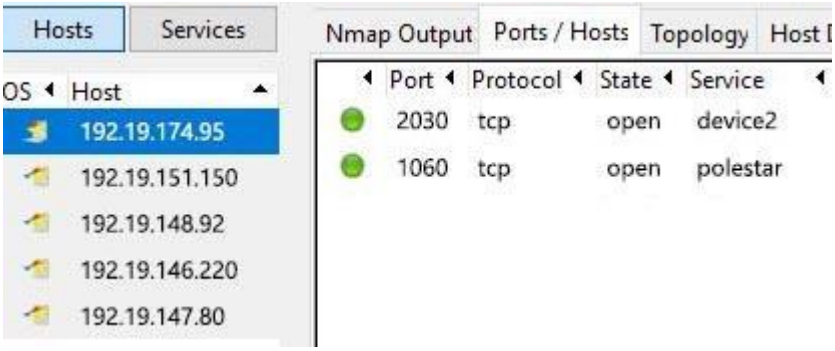

Рис. 4.5. Друге сканування: результат

4.2.3. Дослідження захисту системи використання програмних приманок на основі розробленої моделі від атаки DDoS

Як відомо, DDoS-атака спрямована на марну витрату ресурсів ПК, таким чином не даючи серверу надавати звичайні сервіси та ресурси. Проводиться оцінка продуктивності мережі та час реакції статичних хостів і динамічних серверів (тобто, запропоновану схему) під час атаки SYN DDoS. Тестування атаки здійснюється шляхом безперервної розсилки пакетів SYN з різною швидкістю.

Щоб оцінити продуктивність мережі, розмір пакета SYN для атаки встановлено на 73695 байт у Hping3 v3.2.2, що вказує на те, що пакет розділений на певні пакети TCP. Вимірювання продуктивності мережі здійснюється за допомогою Iperf v3.10.1. Рис. 4.6-4.7 ілюструють вплив швидкості атаки на продуктивність мережі на ефективну пропускну здатність і трафік TCP. Коли швидкість атаки дорівнює 0 (тобто немає атакуючого пакета), обидва типи хостів досягають своїх максимальних значень 736 (Мбайт/с) і 100 (Мбіт/сек) у пропускній здатності TCP і TCP-трафіку відповідно. Однак із збільшенням швидкості атаки відбувається різке уповільнення з 0 до 1000 пакетів в секунду. Мабуть, кут падіння в статичних хостах більший у порівнянні з ламаною лінією динамічних хостів. Як бачимо, спостерігається повільне зростання в діапазоні від 1000 (пакетів/с) до 3000 (пакетів/с), а значення динамічних хостів все ще більше, ніж статичних хостів. Таким чином, динамічна система програмних приманок має перевагу перед стаціонарними хостами з точки зору продуктивності мережі.

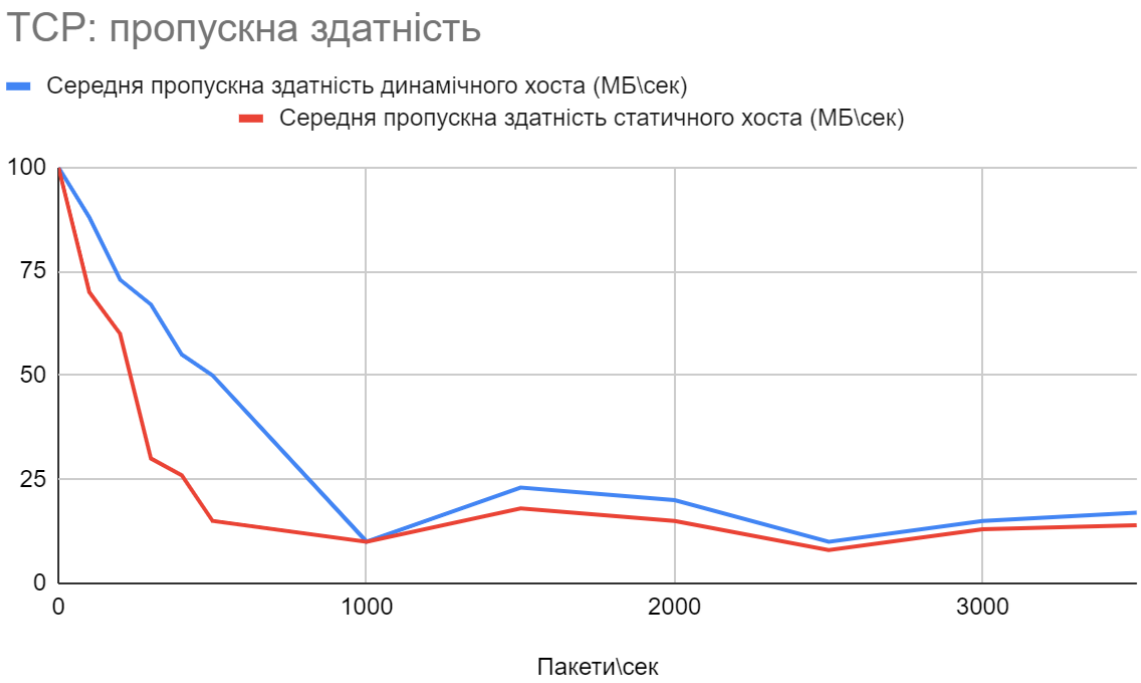

Рис. 4.6. Порівняння пропускної здатності TCP

Щоб відобразити відсоткове відношення між системами, можна порівняти середню пропускну здатність динамічного хоста (ДХ) та статичного хоста (СХ). Щоб порівняти системи, розрахуємо відсоткове відношення між пропускними здатностями ДХ та СХ за формулою: $\sum_n(\left(\frac{DH}{SH}-1\right)*100)/n$, де n= кількість проведених атак, DH = пропускна здатність динамічного хості під час атаки, SH = пропускна здатність статичного хоста під час атаки.

Отже, середнє відсоткове значення становить 54%. Це означає, що динамічний хост (ДХ) в середньому до 54% краще справляється з атаками, ніж статичний хост (СХ).

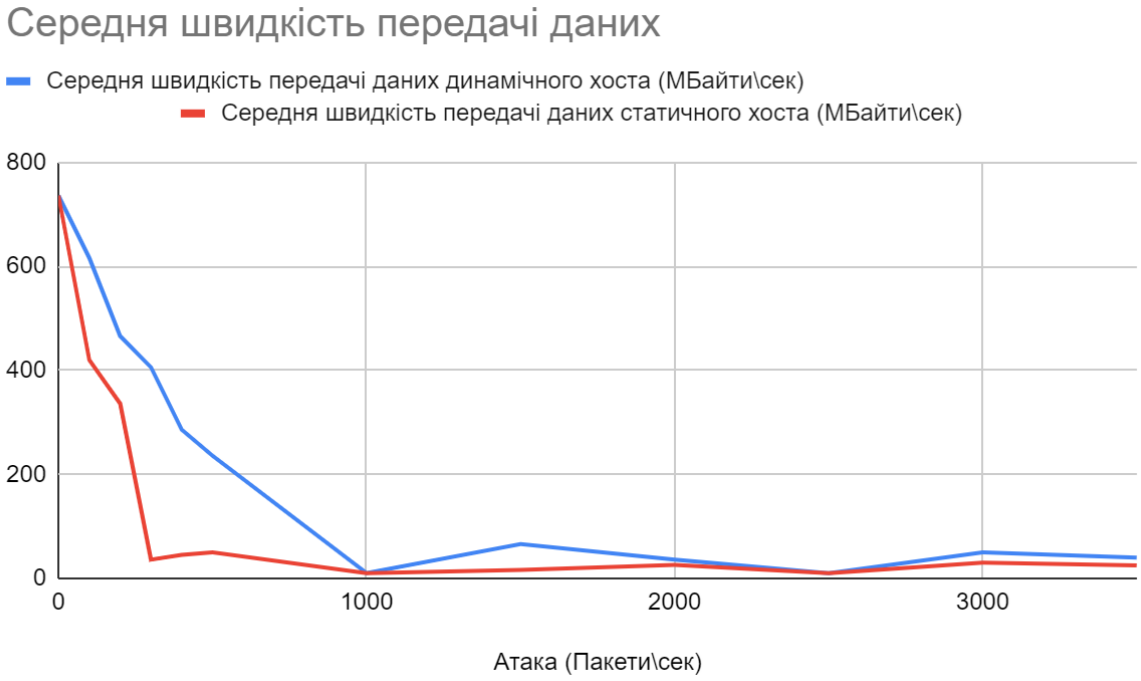

Рис. 4.7. Порівняння середньої швидкості пропускної здатності

Отже, середнє відсоткове значення становить 204%. Це означає, що динамічний хост (ДХ) в середньому до 204% краще справляється з передачею даних під час атак, ніж статичний хост (СХ).

Висновок: на основі цих даних можна зробити висновок, що динамічний хост значно ефективніше передає дані під час атак порівняно зі статичним хостом.

4.2.3.1. Аналіз часу відгуку мережі в системах де використовується розроблена модель та її аналогів

Trafgen в netsniff-ng v0.6.7 використовується для запуску тесту атаки SYN. На відміну від пакета SYN, згаданого в оцінці продуктивності мережі, цей вид пакету складається з 64 байтів для атаки SYN flood. Оскільки в розробленій системі є чотири види сервісів, середній час відповіді послуги стає неодмінним показником оцінки [96]. Вимірювання часу відгуку виконується Jmeter v5.4.2 для кожної служби.

Оператор запиту до бази даних «select * from school» використовується для вимірювання часу отримання відповідних даних. Як показано на рис. 4.8, статичний хост не відповідає зі швидкістю атаки 14 (Кбіт/с). Однак час відгуку

динамічних хостів, здається, залишається незмінним від 0 до 10 (Кбіт/с) по осі X і досягає нескінченного значення після 60 (Кбіт/сек) по осі X. Порівняння з динамічними хостами вражає, тому сервер MySQL статичного хоста страждає від DDoS-атаки. Оскільки п’ять розподілених хостів розподіляють навантаження на атаку, експериментальна крива динамічних хостів демонструє їх перевагу в захисті від DDoS-атаки.

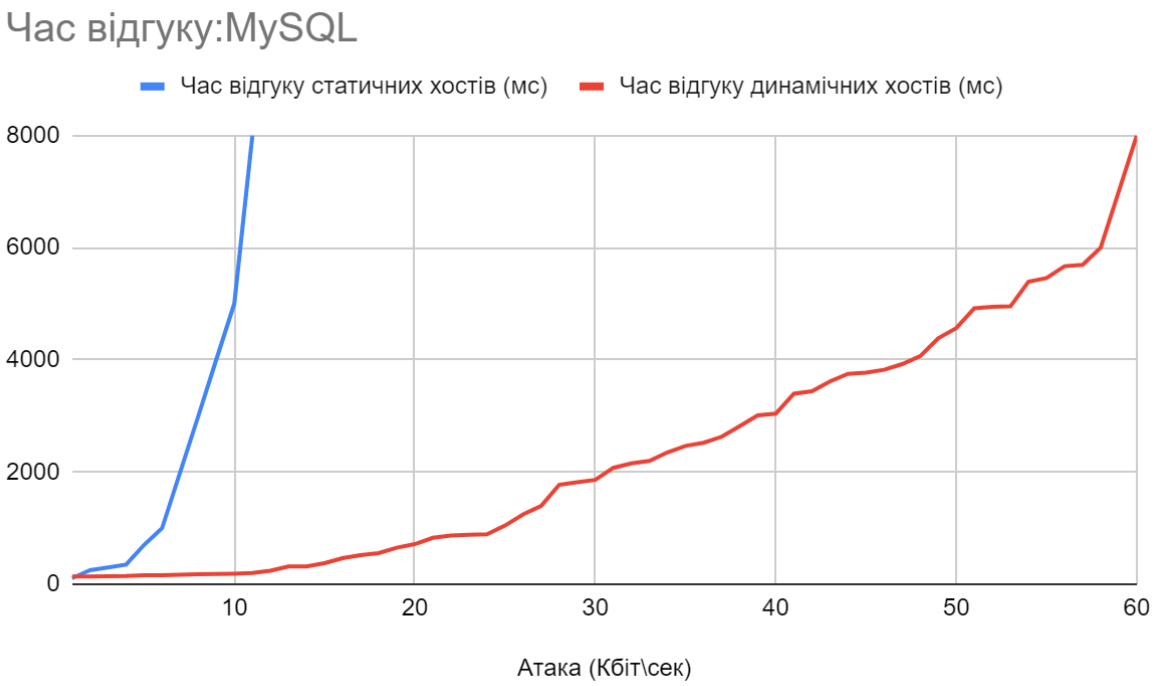

Рис. 4.8. Час відгуку: MySQL

Середнє відсоткове значення становить 34%. Це означає, що статичні хости в середньому до 34% гірше справляються з часом відгуку під час атак, ніж динамічні хости.

Висновок: на основі цих даних можна зробити висновок, що динамічні хости краще справляються з часом відгуку під час атак, ніж статичні хости. Зокрема, динамічні хости в середньому до 34% краще справляються з часом відгуку під час атак.

Перевіряється час завантаження всієї веб-сторінки Apache. На рис. 4.9 динамічні хости витрачають більше часу на завантаження веб-сторінки, ніж статичний хост. Це пояснюється тим, що робота з майнінгу Blockchain виснажує

деякі системні ресурси, що стає ключовим фактором впливу на час відповіді сервера. Час роботи статичного і динамічного хостів майже однакові між собою від 1 (Мбіт/с) до 10 (Мбіт/сек), трохи збільшуючись вздовж осі X. У такому випадку обидва типи хостів піддаються впливу DDoS-атаки. Статичний сервер не відповідає, починаючи з 8,5 (Мбіт/с), а час відповіді динамічного сервера вищий при тій самій швидкості атаки, що вказує на те, що динамічний сервер Apache все ще може реагувати, навіть якщо статичний сервер зазнає збою.

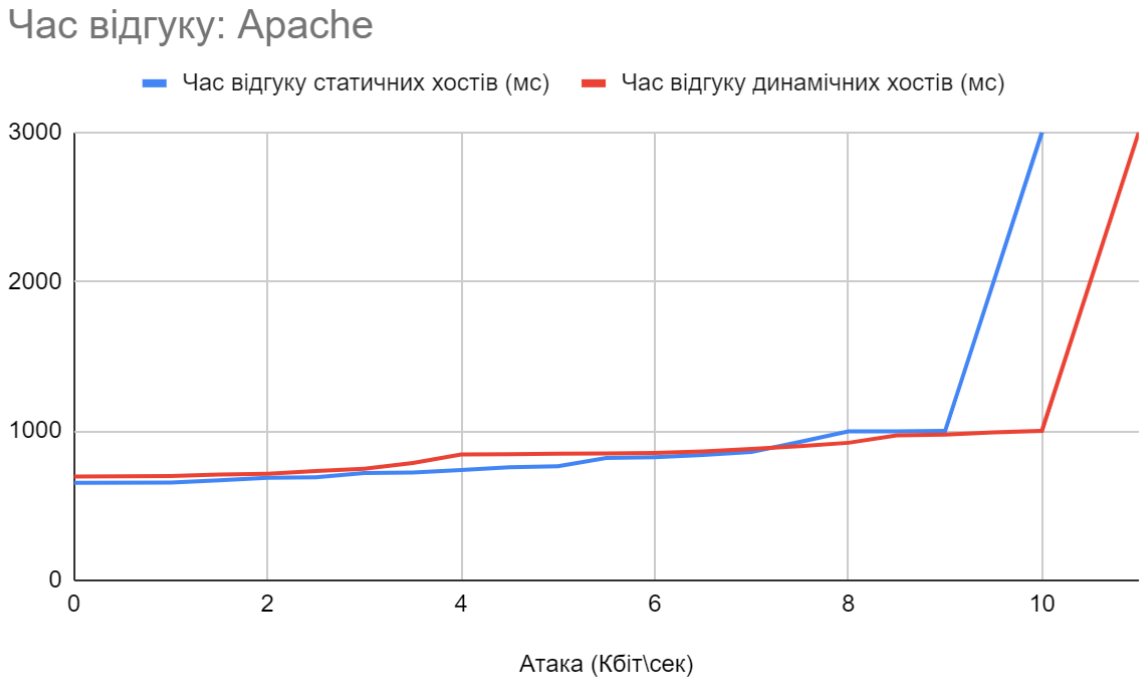

Рис. 4.9. Час відгуку: Apache

Отже, середнє відсоткове значення становить 1%. Це означає, що статичні хости в середньому до 1% гірше справляються з часом відгуку під час атак, ніж динамічні хости.

Висновок: на основі цих даних можна зробити висновок, що динамічні хости краще справляються з часом відгуку під час атак, ніж статичні хости. Різниця складає 1%, тому в реальних сценаріях можуть виникати відмінності залежно від конкретної ситуації.

Криві Vsftpd і Nginx за часом відгуку показані на рис. 4.10-4.11. Час відповіді завантаження txt-файлу з сервера Vsftpd вимірюється під час DDoS-

атаки. На рис. 31 час відгуку Vsftpd у статичному хості стрімко зростає і досягає своєї нескінченності на рівні 11 (Кбіт/сек). Через майнінг Blockchain загальний тренд від 0 до 10 (Кбіт/сек) незначно впливає. Однак рівна тенденція кривої динамічних хостів вказує на стійкість до DDoS-атаки. Оскільки робота з видобутку на динамічних хостах виснажує системні ресурси, Nginx піддається впливу. Як показано на рис. 32, середній час відгуку Nginx у статичному хості перевершує динамічний від 1 до 2,5 (Мбіт/с) вздовж осі X. Після 2 (Мбіт/с) DDoS-атака стає основним фактором, що впливає на час відповіді. Від 2 (Мбіт/с) до 4 (Мбіт/с) крива динамічних хостів завжди нижче за іншу, що означає, що час на статичній хості довший, ніж у динамічних хостах. Відомо, що Nginx створює символи з меншою пам’яттю і високою паралельністю, тому обидві криві зберігають свою м’яку характеристику. Однак час простою статичного сервера відбувається раніше, ніж динамічного, що свідчить про ефективність розробленої схеми.

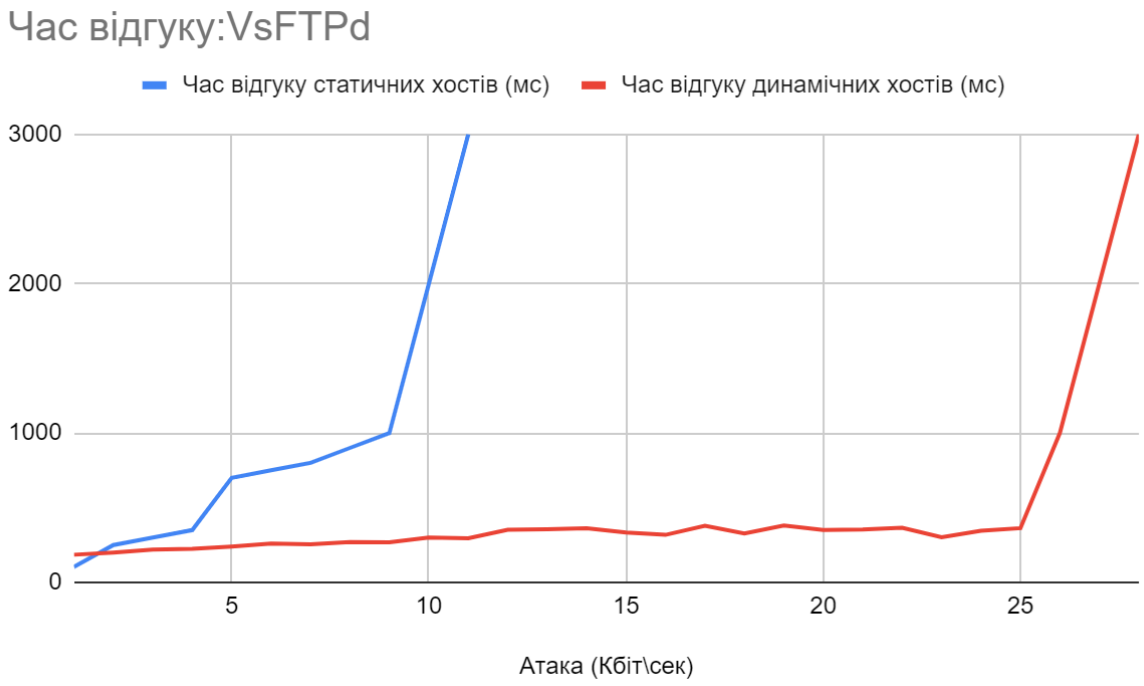

Рис. 4.10. Час відгуку: VsFTPd

Отже, середнє відсоткове значення становить 13%. Це означає, що статичні хости в середньому до 13% гірше справляються з часом відгуку під час атак, ніж динамічні хости.

Висновок: на основі цих даних можна зробити висновок, що динамічні хости краще справляються з часом відгуку під час атак, ніж статичні хости.

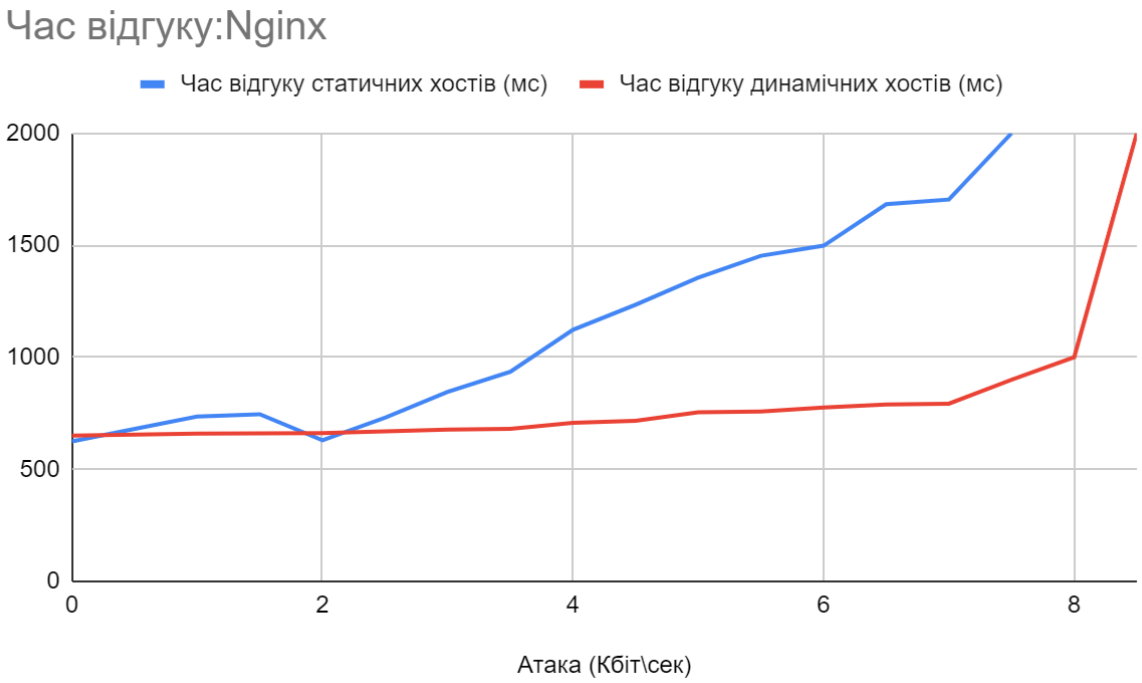

Рис. 4.11. Час відгуку:Nginx

Отже, середнє відсоткове значення становить 16%. Це означає, що статичні хости в середньому до 16% гірше справляються з часом відгуку під час атак, ніж динамічні хости.

Висновок: на основі цих даних можна зробити висновок, що динамічні хости краще справляються з часом відгуку під час атак, ніж статичні хости

Було проведено однакову кількість атак як на централізовану систему, яка використовує програмні приманки так і на динамічну систему, яка побудована на розробленому методі [97]. Розроблений метод використання програмних приманок, що побудований на основі використання технології Blockchain вимагає більше ресурсів від нападника для здійснення атаки на мережу: потужності комп’ютерів, серверів з яких здійснюється атака а також більше фізичного часу,

що збільшує час для фахівців з кібербезпеки для реагування та контр дії нападу до 45%. Найбільшу різницю видно не на всіх сервісах: Apache та Nginx динамічної системи зазнають під час атаки майже однакового з центральним аналогом результатів. Однак сервіси Vsftpd та MySQL вимагають використання значно більших ресурсів від нападника, що показує ефективність розробленого методу у плані захисту комп'ютерної мережі [98]. Порівняння ефективності централізованої моделі від розробленої моделі використання програмних приманок під час атаки зображено на рис. 4.12.

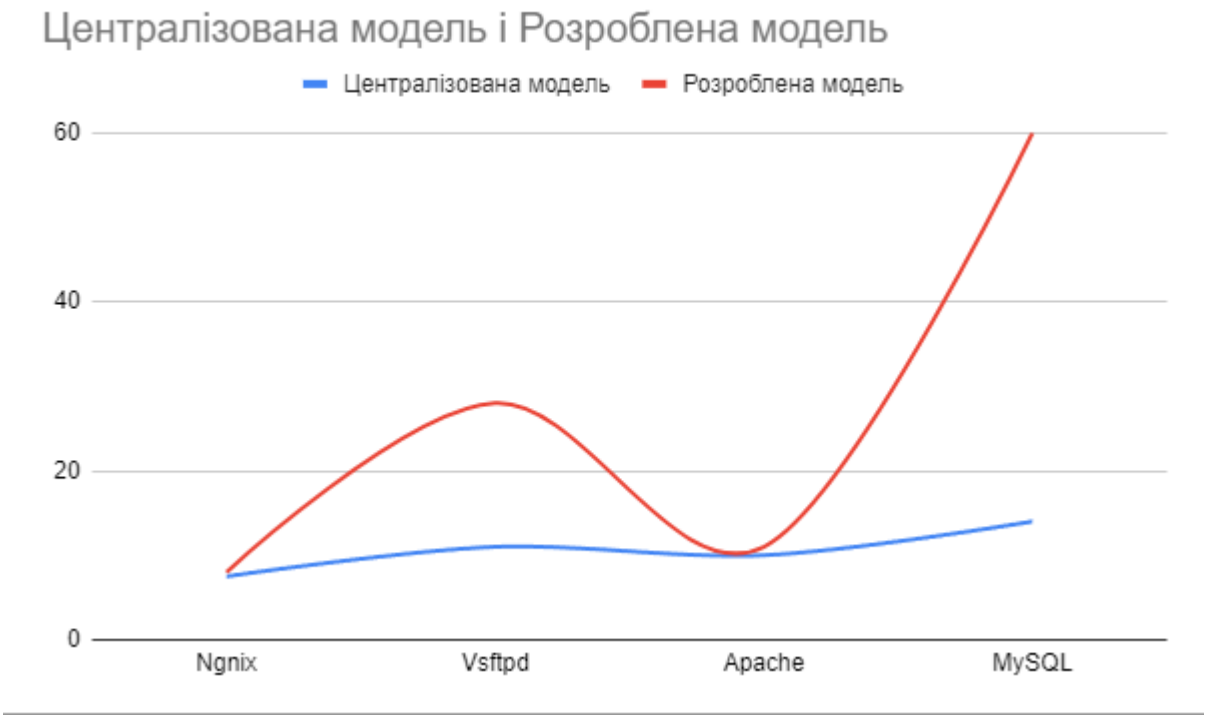

Рис. 4.12. Порівняння ефективності централізованої моделі та розробленої моделі під час атаки

Загальна кількість проведених атак дорівнює 60, по 30 на мережу з розробленою моделлю та з централізованою. Одна атака складалася з трьох різних підходів, які використовувалися раніше та включє в себе атаки DDoS, сніфер та сканування. Атака вважається заблокованою, якщо всі три вразливості були відбиті. На рис. 4.13. зображено результат проведених атак для централізованої моделі. Прогрмані приманки в централізованих моделях не є захисним механізмом, як і згадувалось раніше. Вони слугують попереджувальним шаром в захисній системі і їхнє завдання відволікти зловмисника та зібрати й записати в

журнали атак інформацію про проведені дії. З 30 атак успішно заблоковано по всіх трьох параметрах було всього 4, що складає 13% від загальної кількості. Цей показник має тенденцію зменшуватися зі збільшенням кількості атак. Також видно що 13% атак пройшли повністю по всім параметрам через те, що програмна приманка була знайдена та ігнорована і атака продовжувалась одразу на легітимну мережу. DDoS атака була заблокована 11 раз з 30, що складає 36% атак даного типу, атака сканування заблокована 17 раз з 30, що складає 56% атак даного типу, сніфер атака заблокована 19 раз з 30, що складає 63% атак даного типу.

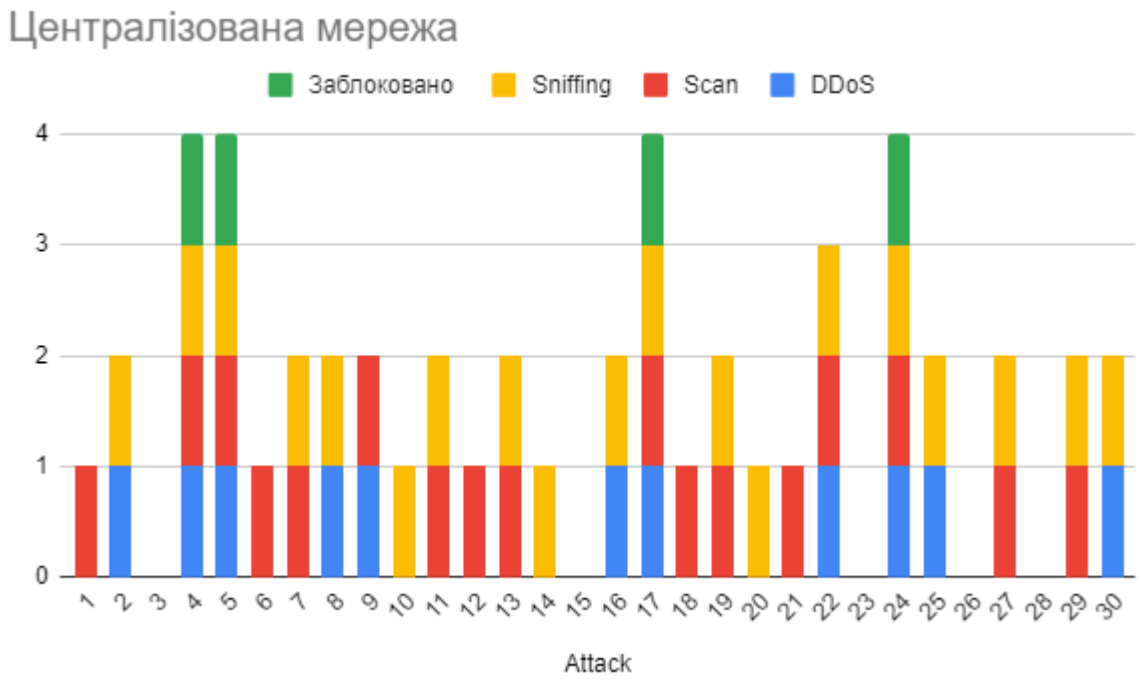

Рис. 4.13. Результат атак на централізовану модель

На рис. 4.14. зображено результат проведених атак для розробленої моделі. З 30 атак успішно заблоковано половину з них, що складає 50% від загальної кількості. Атаки сканування та сніфер неможливо було успішно провести через транслокаційні можливості розробленої мережі, динамічними змінами хостів та постійне потрапляття на програмну приманку. Найслабшою ланкою мережі виявляються DDoS атаки, статистика яких вплинула на загальну картину. Підвищення рівня пропускної здатності та часу відгуку сервісів прямо пропорційно впливає на цей результат, оскільки від зловмисника вимагаються

більші обчислювальні потужності для проведення атаки. Так як розроблена мережа складається всього з п’яти вузлів (п’ять персональних комп’ютерів) то й відповідно навантаження на них більше. Зі збільшенням кількості вузлів в мережі навантаження буде більш розподіленим та результати можуть покращитися в декілька разів. DDoS атака була заблокована 15 раз з 30, що складає 50% атак даного типу, атака сканування та сніфер атака заблоковані у всіх тридцяти атаках, що складає 100% атак даних типів. Однак з рис. 4.14. видно, що програмні приманки, побудовані на основі технології Blockchain слугують повноцінним захисним механізмом та виконують не лише моніторингові задачі а й задачі безпосереднього захисту комп’ютерної мережі, що підвищує загальний рівень захисту мережі в порівнянні з аналогами.

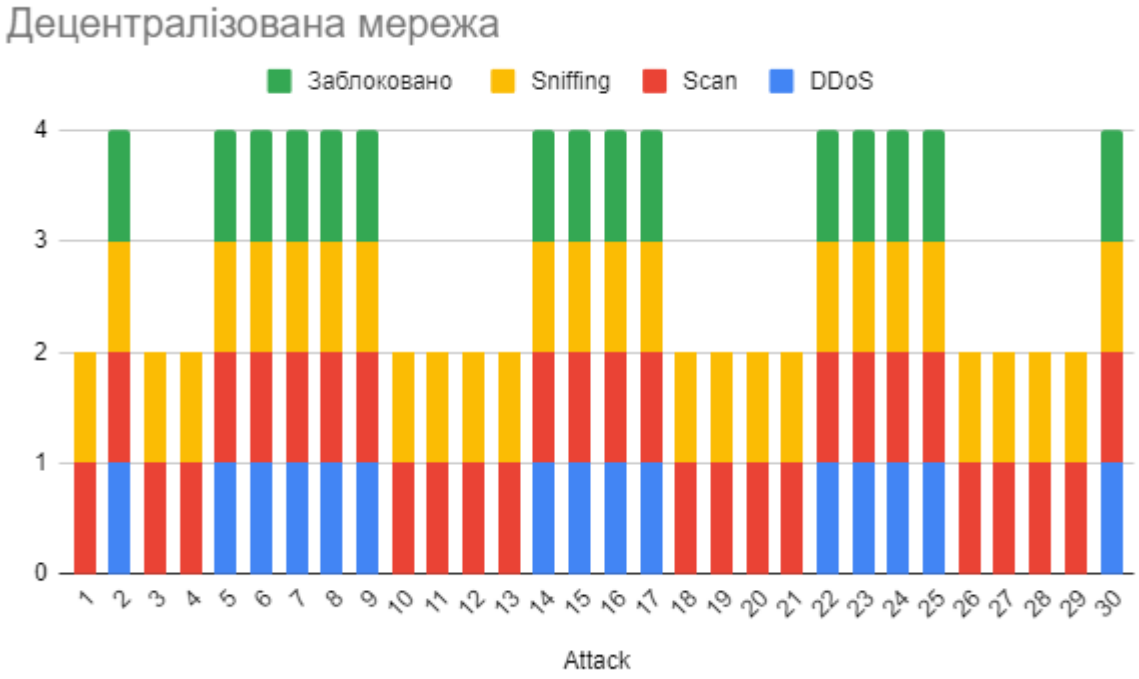

Рис. 4.14. Результат атак на розроблену модель

Порівнюючи централізовану модель з децентралізованою отримуємо наступні результати:

1. Захист децентралізованої моделі мережі під час DDoS атаки вищий на 14%.
2. Захист децентралізованої моделі мережі під час атаки сканування вищий на 44%. Майже в два рази.
3. Захист децентралізованої моделі мережі під час сніфер атаки вищий на 37%.

4. Загальний захист комп’ютерної мережі побудованої з використанням прогрманих приманок на основі технології Blockchain вищий на 37% в порівнянні з централізованим аналогом, що є підвищенням глобального рівня захисту комп’ютерної мережі в півтора рази.

4.3. Порівняльний аналіз розробленого динамічного методу з статичними аналогами

Щоб краще проілюструвати захисні можливості розробленого методу, час відповіді чотирьох служб без будь-якої атаки зведено в таблицю 4.1.

Таблиця 4.1.

Швидкість відгуку сервісів, коли немає жодної атаки.

Сервіс	Статичний хост	Динамічний хост
Ngnix	625 мс	650 мс
Vsftpd	103 мс	185 мс
Apache	653 мс	695 мс
MySQL	105 мс	135 мс

Часи на статичному хості завжди менші, ніж у динамічного. Навіть якщо значення складності у файлі genesis коригується на «0x400» для зменшення накладних витрат на обчислення, механізм консенсусу для генерації блоків робить динамічні сервіси менш ефективними за часом відповіді [99]. Тим не менш, кінцеві швидкості атаки динамічних хостів, які призводять до збою служби, показують перевагу над статичним хостом, як показано в таблиці 4.2.

Таблиця 4.2.

Швидкість атаки перед тим, як сервіс перестає приймати запити

	Статичний хост	Динамічний хост
Ngnix	7.5 Мб\с	8.0 Мб\с

Vsftpd	11 Кб\с	28 Кб\с
Apache	10 Мб\с	11 Мб\с
MySQL	14 Кб\с	60 Кб\с

Без трафіку атаки низька ефективність динамічних хостів призводить до більш тривалого часу відповіді. Однак із збільшенням трафіку атак на динамічні хости вони демонструють переваги в захисті системи, що свідчить про ефективність запропонованої схеми.

4.3.1. Порівняння методу використання програмних приманок побудованих з використанням технології Blockchain як елементів захисту з іншими рішеннями

Для порівняння розробленої схеми з іншими варіантами, запропонованими для динамічного Blockchain, представлені три динамічні роботи, пов’язані з Blockchainom. Blockchain використовується для динамічного керування ключами при використанні динамічних програмних приманок, де сторонні повноваження видаляються, а функції центрального менеджера об’єднуються в мережу менеджера безпеки. Протокол перевірки для динамічного доступу до спектру, заснований на Blockchain, виконується в гібридних приманках, а центральний уповноважений вузол видалено [100]. Blockchain в основному використовується для запису транзакцій. У адаптивних приманок Blockchain прийнято до динамічної розподіленої схеми контролю доступу для пристроїв, що автентифікуються IoT, і дозвіл доступу зберігається в мережі Blockchain. Динамічна політика створюється за допомогою смарт-контракту для пристрою IoT, який зареєстровано без асоційованих політик. У цих роботах в основному використовується незмінний, розподілений і децентралізований характер Blockchain, який схожий на розроблену схему. Порівняння з іншими суміжними рішеннями наведено в таблиці 4.3.

Таблиця 4.3.

Порівняння розробленого прототипу системи з іншими рішеннями на ринку

ПП	Динамічна ПП	Гібридна ПП	ПП	Адаптивна ПП	Розроблена модель
Багатопоточність	Так	Так	Так	Так	Так
Розподілена архітектура	Так	Так	Так	Так	Так
Децентралізація					Так
Система анти-модифікації					Так
Розгортання	Так	Так		Так	Так
Конфігурація		Так	Так	Так	Так
Об’єкт	Програмні приманки	Програмні приманки	Програмні приманки	Програмні приманки	Програмні приманки та справжні сервіси
Сценарій	Система виявлення вторгнень	Система виявлення вторгнень	Мережа	Мережа	Мережа

Завдання	Виявлення вторгнень	Покращення системи виявлення вторгнень	Пастка	Авто розгортання	Захист мережевих систем

Усі ці динамічні схеми (Звичайна ПП (програмна приманка), динамічна ПП, гібридна ПП, адаптивна ПП) оснащені мультимедіями. Однак їх розподілені архітектури контролюються центральною стороною, що не підтверджує властивість децентралізації. Проблема централізації призводить до єдиної точки збою, що ще більше призводить до збою всієї системи. Крім того, дані, що зберігаються в цій базі даних, можуть бути змінені в небезпечних умовах. За умовою розроблена схема є децентралізованою та незмінною на основі Blockchain. Динамічна інформація цих схем відображається в розгортанні або конфігурації. Однак ця інформація стосується лише приманок, а не справжньої системи. Після того, як ці приманки розпізнаються, фіксована реальна система стикається з загрозою нападу [101]. Таким чином, динамічні об’єкти розробленої схеми містять як приманки, так і реальні сервіси. Періодично змінюючи реальні сервіси, зловмисник не може їх знайти, тим самим захищаючи систему. Перш за все, порівняння показує переваги розробленої системи схеми.

Висновки до розділу 4

Таким чином, у четвертому розділі дисертаційної роботи було проведено порівняльний аналіз побудованої системи на основі технології Blockchain яка використовує програмні приманки на основі розробленого методу з динамічними атрибутами та аналогами цих систем для перевірки рівня захищеності даної мережі. Було проведено атаки типу DDoS, сніфер та сканування на розроблену систему та аналоги. Проведене дослідження дозволяє виділити наступне:

1. У підсумку, усі експериментальні результати показують, що запропонована динамічна система програмних приманок перевершує традиційну фіксовану

систему. Оскільки Blockchain споживає системні ресурси, експериментальні дані незначно впливають на динамічні хости.

2. Розроблений метод інтегрує децентралізовані та автоматично оновлювані атрибути приманок, що дало змогу підвищити ефективність захисту мережі шляхом зменшення навантаження на мережеву інфраструктуру та часу відгуку сервісів у разі атаки на 54% підвищити пропускну здатність каналу та до 204% підвищити швидкість передачі даних під час проведення зовнішніх атак на систему в порівнянні з статичними аналогами, а вдосконалений математичний опис обчислення динамічних атрибутів програмних приманок дав змогу покращити час відгуку сервісів на які проводиться атака типу DDoS, а саме: MSQL до 34%, NGNIX до 16%, APACHE до 1%, vsFTPd до 13% в порівнянні з результатами отриманими під час експериментів зі статичними аналогами.

3. Розроблений метод використання програмних приманок, що побудовані на основі використання технології Blockchain зменшує навантаження на всю систему на відміну від традиційного фіксованого рішення. Завдяки оцінці продуктивності вище, можна стверджувати, що загальний час відгуку та продуктивність мережі розробленої схеми мають переваги над традиційною системою.

4. Розроблений метод використання програмних приманок, що побудований на основі використання технології Blockchain вимагає більше ресурсів від нападника для здійснення атаки на мережу: потужності комп’ютерів, серверів з яких здійснюється атака а також більше фізичного часу, що збільшує час для фахівців з кібербезпеки для реагування та контр дії нападу до 45%.

5. Розроблений метод використання програмних приманок в порівнянні з централізованим аналогом пропонує підвищення рівня захисту комп’ютерної мережі. З тридцяти проведених атак розроблена модель успішно заблокувала 50% в той час як централізована всього 13%, що показує покращення захисних можливостей розробленої моделі.

ВИСНОВКИ

В роботі вирішено важливу науково-практичну проблему, а саме підвищення ефективності виявлення кіберзлочинів та покращення стійкості захисної системи за рахунок розробки системи приманок на основі динамічних атрибутів Blockchain.

У підсумку, усі експериментальні результати показують, що запропонована динамічна система програмних приманок перевершує традиційну фіксовану систему. Оскільки Blockchain споживає системні ресурси, експериментальні дані незначно впливають на динамічні хости. Проте розроблена схема зменшує навантаження на відміну від традиційного фіксованого рішення.

1. Проведено огляд існуючих рішень та реалізації програмних приманок, а також проаналізовані властивності Blockchain технології для виявлення точок перетину технологій, які можна було б використати для підсилення системи захисту. На основі проведеного аналізу встановлено, що відомі методи використання програмних приманок не мають потрібної гнучкості при управлінні та реакції на зовнішні атаки. Обгрунтовано актуальність науково-практичного завдання дисертаційного дослідження. Зокрема встановлено, що для підсилення захисту комерційних та державних кіберсистем в умовах розвитку технологій та методів атак зровмисниками є розроблення нових, адаптивних методів захисту.

2. Запропоновано удосконалення класичного алгоритму визначення та передачі вузлових хостів (за рахунок плаваючих хостів в мережі) в системі Blockchain, що дозволило підвищити загальну адаптивність мережі реагувати на зовнішні атаки. Цей алгоритм дозволяє системі реагувати на атаки типу сканування та закривати порти доступу реагуючи на зловмисні дії. Під час атаки сканування відкриті порти закриваються (за рахунок зміни основного хоста), що ускладнює збір інформації та можливість доступитись до системи ззовні, а відтак отримано перевагу в захисті всієї мережі.

3. Розроблена модель динамічної системи активних пасток на основі програмних приманок та розробленого методу їх використання, що використовують Blockchain технологію. Дана модель інтегрує децентралізовані та

автоматично оновлювані атрибути пасток, що дало змогу підвищити ефективність захисту мережі шляхом зменшення навантаження на мережеву інфраструктуру та часу відгуку сервісів у разі атаки до 54% підвищити пропускну здатність каналу та до 204% підвищити швидкість передачі даних під час проведення зовнішніх атак на систему в порівнянні з статичними аналогами. Це збільшує час реагування ланок-валідаторів на атаку та дозволяє прийняти ефективне безпекове рішення, наприклад ізоляція ланки та видалення її з мережі, перед тим як впасти, що підвищує стійкість та глобальний захист мережі.

4. Розроблений метод використання програмних приманок, що побудований на основі використання технології Blockchain вимагає більше ресурсів від нападника для здійснення атаки на мережу: потужності комп’ютерів, серверів з яких здійснюється атака а також більше фізичного часу, що збільшує час для фахівців з кібербезпеки для реагування та контр дії нападу до 45%. Найбільшу різницю видно не на всіх сервісах: Apache та Nginx динамічної системи зазнають під час атаки майже однакового з центральним аналогом результатів. Однак сервіси Vsftpd та MySQL вимагають використання значно більших ресурсів від нападника, що показує ефективність розробленого методу у плані захисту.

5. Розвинуто математичний опис обчислення динамічних атрибутів програмних приманок, який враховує динамічні та транс локаційні можливості Blockchain-технології Solana. Це дало змогу змоделювати та оптимізувати розподіл ресурсів мережі за рахунок адаптації до змінних умов, що в результаті сприяло підвищенню ефективності захисту, зокрема забезпеченню швидкого відгуку сервісів під час зовнішніх атак. Зокрема вдосконалений математичний опис обчислення динамічних атрибутів програмних приманок дав змогу покращити час відгуку сервісів на які проводиться атака типу DDoS, а саме: MSQL до 34%, NGNIX до 16%, APACHE до 1%, vsFTPd до 13% в порівнянні з результатами отриманими під час експериментів зі статичними аналогами. Розроблений метод використання програмних приманок, що побудований на основі використання технології Blockchain вимагає більше ресурсів від нападника для здійснення атаки на мережу: потужності комп’ютерів, серверів з яких

здійснюється атака а також більше фізичного часу, що збільшує час для фахівців з кібербезпеки для реагування та контр дії нападу.

6. Розроблений метод динамічної системи програмних приманок, який використовує Blockchain-технологію для підтримки безпеки, прозорості та адаптивності до зовнішніх атак, за рахунок плаваючих хостів в мережі. Цей метод, на відміну від відомих унеможливлює здійснення успішної сніфер атаки за рахунок шифрування, захищає від атаки сканування за рахунок динамічного відкривання та закривання портів, підвищує ефективність захищеності від DDoS атак та збергіє інформацію про атаки на систему на Blockchain-платформі, що забезпечує високий рівень збереження даних та гарантує їхню незмінність. Метод включає шифрування RSA 2048 - бітовим алгоритмом шифрування, який неможливо декодувати без відповідного ключа конфіденційності, що забезпечує захист каналу зв'язку та запобігає витоку даних через перехоплення та розшифрування інформації під час передачі даних. Експеримент з сніфер атакою на розроблену модель системи показує, що перехоплені дані неможливо розшифрувати. Розвитком даного дослідження може стати покращення системи динамічної зміни хоста та алгоритмів закривання-відкривання портів.

7. Розроблений метод використання динамічних програмних приманок побудованих на основі технології Blockchain показує кращі показники в порівнянні зі статичними та динамічними аналогами. Середня швидкість пропускної здатності хостів більша до 204%, а за характеристиками стійкості та швидкості відгуку сервісів під час зовнішніх атак значення показників коливаються в залежності від конкретного навантаженого сервісу в межах 15%. З Усі динамічні схеми (Звичайна ПП (програмна приманка), динамічна ПП, гібридна ПП, адаптивна ПП) оснащені мультимедіями. Однак їх розподілені архітектури контролюються центральною стороною. Проблема централізації призводить до єдиної точки збою, що ще більше призводить до збою всієї системи. Крім того, дані, що зберігаються в цій базі даних, можуть бути змінені в небезпечних умовах. За умовою розроблена схема є децентралізованою та незмінною на основі Blockchain. Динамічна інформація цих схем відображається в

розгортанні або конфігурації. Динамічні об’єкти розробленого методу містять як приманки, так і реальні сервіси. Періодично змінюючи реальні сервіси, зловмисник не може їх знайти, тим самим захищаючи систему. Підвищення пропускної здатності та часу відгуку сервісів в комплексі з захисними механізмами технології Blockchain та використанням розробленого методу використання програмних приманок з динамічними атрибутами та мігруючими хостами показує, що отримано перевагу в захисті комп’ютерної мережі.

8. Розроблений метод використання програмних приманок в порівнянні з централізованим аналогом пропонує підвищення рівня захисту комп’ютерної мережі. З тридцяти проведених атак розроблена модель успішно заблокувала 50% в той час як централізована всього 13%, що показує покращення захисних можливостей розробленої моделі. Найслабшою ланкою мережі виявляються DDoS атаки, статистика яких вплинула на загальну картину. Підвищення рівня пропускної здатності та часу відгуку сервісів прямо пропорційно впливає на цей результат, оскільки від зловмисника вимагаються більші обчислювальні потужності для проведення атаки. Так як розроблена мережа складається всього з п’яти вузлів (п’ять персональних комп’ютерів) то й відповідно навантаження на них більше. Зі збільшенням кількості вузлів в мережі навантаження буде більш розподіленим та результати можуть покращитися в декілька разів.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. M. Anirudh, S. A. Thileeban, and D. J. Nallathambi, ‘‘Use of honeypots for mitigating DoS attacks targeted on IoT networks,’’ in Proc. Int. Conf. Comput., Commun. Signal Process. (ICCCSP), Jan. 2017, pp. 1–4.

2. S. Anjali and J. Ramesh, ‘‘An auto-responsive honeypot architecture for dynamic resource allocation and QoS adaptation in DDoS attacked networks,’’ Comput. Commun., vol. 32, no. 12, pp. 1384–1399, Jul. 2009

3. Kuwatly, M. Sraj, Z. Al. Masri, and H. Artail, ‘‘A dynamic honeypot design for intrusion detection,’’ in Proc. IEEE/ACS Int. Conf. Pervas. Services, Jul. 2004, pp. 95–104.

4. Hassan, S. Haidar, S. Malek, K. Iyad, and A. M. Zaid, ‘‘A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks,’’ Comput. Secur., vol. 25, no. 4, pp. 274–288, Jun. 2006.

5. Saeedi, H. Khotanlou, and M. Nassiri, ‘‘A dynamic approach for honeypot management,’’ Int. J. Inf., Secur. Syst. Manage., vol. 1, no. 2, pp. 104–109, Dec. 2012.

6. W. Fan, D. Fernandez, and Z. Du, ‘‘Adaptive and flexible virtual honeynet,’’ in Proc. Int. Conf. Mobile, Secure Program. Netw., 2015, pp. 1–17.

7. Hecker and B. Hay, ‘‘Automated honeynet deployment for dynamic network environment,’’ in Proc. 46th Hawaii Int. Conf. Syst. Sci., Jan. 2013, pp. 4880–4889.

8. Fraunholz, M. Zimmermann, and H. D. Schotten, ‘‘An adaptive honeypot configuration, deployment and maintenance strategy,’’ in Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT), Feb. 2017, pp. 53–57.

9. W. Fan, D. Fernńdez, and Z. Du, ‘‘Versatile virtual honeynet management framework,’’ IET Inf. Secur., vol. 11, no. 1, pp. 38–45, 2016.

10. Casino, F., Dasaklis, T. K., and Patsakis, C. (2018). A systematic literature review of Blockchain-based applications: current status, classification and open issues. Telemat. Informat. 36, 55–81. doi: 10.1016/j.tele.2018.11.006

11. Hepp, T., Wortner, P., Schönhals, A., and Gipp, B. (2018). “Securing physical assets on the Blockchain: linking a novel object identification concept with distributed

ledgers,” in Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock '18) (Munich: ACM), 60–65. doi: 10.1145/3211933.3211944

12. Cruz, J. P., Kaji, Y., and Yanai, N. (2018). RBAC-SC: role-based access control using smart contract. IEEE Access 6, 12240–12251. doi: 10.1109/ACCESS.2018.2812844

13. Swan, M. (2015b). Blockchain thinking: the brain as a decentralized autonomous corporation. IEEE Technol. Soc. Mag. 34, 41–52. doi: 10.1109/MTS.2015.2494358

14. Schütte, J., Fridgen, G., Prinz, W., Rose, T., Urbach, N., Hoeren, T., et al. (2018). Blockchain ans Smart Contracts - Technologies, Research Issues and Applications. Fraunhofer Society. Available online at: https://bit.ly/2PQ9oI5

15. Susukailo, V., Vasylyshyn, S., Opirskyy, I., Buriachok, V., & Riabchun, O. (2021). Cybercrimes investigation via honeypots in cloud environments. Paper presented at the CEUR Workshop Proceedings, 2923 91-96.

16.С. Василишин, І. Опірський, А. Піскозуб, Аналіз використання програмних приманок як засобу забезпечення інформаційної безпеки, DOI: 10.28925/2663-4023.2020.10.8897

17. Валерій Дудикевич, Іван Прокопишин, Василь Чекурін, Іван Опірський, Юрій Лах, Тарас Крет, Євгенія Іванченко, Ігор Іванченко. Багатокритеріальний аналіз ефективності консервативних систем захисту інформації // Східноєвропейський журнал технологій підприємства. Інформаційно-керуюча система. – Т. 3, № 9(99), С. 6-13, (2019). https://doi.org/10.15587/1729-4061.2019.166349

18. Banafa, A. (2017, August 17). Як захистити Інтернет речей (IoT) за допомогою Blockchain. [Онлайн]. Доступно: https://datafloq.com/read/securing-internet-ofthings-iot-with-Blockchain/2228.

19.PwC 2018 Global Economic Crime and Fraud Survey (2018). [Онлайн]. Доступно: https://www.pwc.com/gx/en/news-room/docs/pwc-global-economic-crime-survey-report.pdf

20. Маклафлін, Марк-Девід і Дженіс Гоган. «Проблеми та передовий досвід

управління інформаційною безпекою». MIS Quarterly Executive 17.3 (2018): 237-262.

21. Joshi, RC & Sardana, A. 2011, "Honeypots: нова парадигма інформаційної безпеки" в Honeypots: A New Paradigm to Information Security, стор. 1-323.

22. Журавчак Д. (2021) Створення системи запобігання поширенню вірусів-екстракторів з використанням мови програмування python та утиліти auditd на базі операційної системи. Електронне фахове наукове видання «Кібербезпека: освіта, наука, технології», 4 (12), 108-116. h ttps://doi.org/10.28925/2663-4023.2021.12.108116

23. Gandotra, V., Singhal, A., & Bedi, P. (2012). Концепція безпеки, орієнтована на загрози: проактивний підхід до управління загрозами. Procedia Technology, 4, 487-494. DOI:10.1016/j.protcy .2012.05.078

24. Onaolapo, J., Mariconti, E., & Stringhini, G. (2016). Що відбувається після того, як ви Pwnd: розуміння використання витоку облікових даних облікового запису в дикій природі . Матеріали 2016 ACM on Internet Measurement Conference - IMC 16. DOI:10.1145/2987443.2987475

25. Bamert, T., Decker, C., Elsen, L., Wattenhofer, R., & Welten, S. (2013). Перекусити, розрахуватися біткоінами (pp. 1-5). IEEE. https://doi.org/10.1109/P2P.2013.6688717

26. PwC 2020 Як Blockchain може трансформувати оборонні активи та дати збройним силам перевагу на полі бою [Онлайн]. Доступно: https://www.pwc.com/gx/en/aerospace-defence/pdf/Blockchain-defence.pdf

27. Beecroft, N. (2015). Звіт про нові ризики Bitcon за 2015 рік. [Онлайн]. Доступно: https://www.lloyds.com/~/media/files/news-and-insight/risk.../2015/bitcoin--final.pdf.

28. Bentov, I., Lee, C., Mizrahi, A., & Rosenfeld, M. (2014). Підтвердження активності: розширення підтвердження роботи біткойна за допомогою підтвердження частки (No. 452). [Онлайн]. Доступно: http://eprint.iacr.org/2014/452.

29. Викрадення BGP. (2018, Січень 16). Вікіпедія. [Онлайн]. Доступно:

https://en.wikipedia.org/w/index.php?title=BGP_hijacking&oldid=820773357.

30. Bissias, G., Ozisik, A. P., Levine, B. N., & Liberatore, M. (2014). Стійке до Sybil змішування для біткойнів (pp. 149-158). ACM Press. https://doi.org/10.1145/2665943.2665955

31. Зворотний відлік зменшення винагороди за блок біткойн вдвічі. [Онлайн]. Доступно: http://www.bitcoinblockhalf.com/.

32. Діаграми та графіки біткойн–Blockchain. [Онлайн]. Доступно: https://Blockchain.info/charts.

33. Індекс енергоспоживання біткойнів.[Онлайн]. Доступно: https://digiconomist.net/bitcoin-energy-consumption.

34. Статистика Bitcoin, Litecoin, Namecoin, Dogecoin, Peercoin, Ethereum. [Онлайн]. Доступно: https://bitinfocharts.com/.

35. Розробка Bitcoincore – Скільки BIP 62 («Робота з пластичністю») було реалізовано? – Обмін стеками біткойнів. [Онлайн]. Доступно: https://bitcoin.stackexchange.com/questions/35904/how-much-of-bip-62-dealing-withmalleability-has-been-implemented.

36.Blockchain та технології розподілених нод. [Онлайн]. Доступно: https://www.geeksforgeeks.org/Blockchain-and-distributed-ledger-technology-dlt/

37. Bonneau, J. Навіщо купувати, коли можна орендувати? Хабарні атаки на консенсус у стилі біткойн (p.8). Stanford University and Electronic Frontier Foundation.

38. Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., & Wustrow, E. (2013). Еліптична криптографія на практиці (No. 734). [Онлайн]. Доступно: http://eprint.iacr.org/2013/734.

39. Boverman, A. (2011, Травень 25). Culubas: Timejacking & Bitcoin. [Онлайн]. Доступно: http://culubas.blogspot.com/2011/05/timejacking-bitcoin_802.html.

40. Bruce, J. D. Схема міні-Blockchain, p. 13. [Онлайн]. Доступно: https://www.weusecoins.com/assets/pdf/library/The%20Mini-Blockchain%20Scheme.pdf.

41. Buldas, A., Kroonmaa, A., & Laanoja, R. (2013). Інфраструктура безключових

підписів: як створити глобальні розподілені хеш-дерева (No. 834). [Онлайн]. Доступно: http://eprint.iacr.org/2013/834.

42. Buterin, V. (2014). Смарт-контракт наступного покоління та децентралізована платформа додатків.

43.[Case Study] Атака 51% і подвійні витрати. (2018, Липень 6). [Онлайн]. Доступно: https://www.coindesk.com/markets/2018/06/08/Blockchains-once-feared-51-attack-is-now-becoming-regular/

44. Castro, M., & Liskov, B. (n.d.). Практична візантійська помилковість. У матеріалах третього симпозіуму з проектування та впровадження операційних систем, New Orleans, USA, Лютий 1999. [Онлайн]. Доступно: http://pmg.csail.mit.edu/papers/osdi99.pdf.

45. Chen, T., Li, X., Luo, X., & Zhang, X. (2017). Недостатньо оптимізовані смарт-контракти пожирають ваші гроші. In 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 442-446). https://doi.org/10.1109/SANER.2017.7884650

46. Kaur, P., & Sharma, A. (2018). Honeypot technology: A comprehensive survey. Computers & Security, 77, 445-478.

47. Alharbi, H., & Alqahtani, S. (2019). A review of honeypots: Technologies, threats, and tools. International Journal of Computer Applications, 179(41), 20-25.

48. Xing, H., Li, X., Li, H., Li, Z., & Li, X. (2018). Design and implementation of a scalable honeynet based on container virtualization technology. IEEE Access, 6, 40704-40714.

49. Han, J., Zhang, T., Jia, Y., & Wu, Y. (2019). A novel honeypot-based intrusion detection method in smart grid environment. IEEE Access, 7, 3788-3797.

50. Ray, P. P., & Chakraborty, S. (2019). Honeypots: A comprehensive survey. Journal of Ambient Intelligence and Humanized Computing, 10(8), 3307-3335.

51. Sengupta, S., & Basu, A. (2017). Honeypots: An overview. International Journal of Computer Science and Mobile Computing, 6(6), 129-134.

52. Swami, R., & Bera, S. (2019). Honeypot technology and its role in cyber security. Journal of Information Security and Cybercrimes, 1(1), 6-11.

53. Dai, Y., & Bai, X. (2020). Research on honeypot technology and its application in network security. Journal of Physics: Conference Series, 1469(1), 012074.

54. De Luca, A., & Ferrara, F. (2017). A novel honeypot architecture for web applications security. IEEE Access, 5, 3893-3904.

55. Shukla, S., Yadav, S. K., & Patel, R. B. (2019). A comprehensive review on honeypot: A tool for cyber security. International Journal of Advanced Research in Computer Science and Software Engineering, 9(4), 295-299.

56. Kang, J., Yoo, S., & Kim, C. (2019). A study on the design of a honeypot-based intrusion detection system. Journal of Internet Technology, 20(6), 1823-1833.

57. Sultana, S., & Biswas, G. (2017). Honeypots: A review on cyber security system. International Journal of Computer Applications, 163(2), 1-6.

58. Ahmed, H. M., Alsalibi, B., & Aljahdali, S. (2021). Honeypot technology: Classification, architecture, and open challenges. IEEE Access, 9, 45147-45174.

59. Qazi, Z. A., & Ibrahim, R. (2017). A review on honeypot technology and its different types of deployment. Journal of Information Security Research, 8(2), 34-42.

60. Daniel, J., Rao, N. S. V., & Rai, B. G. (2016). A Comprehensive Study on Honeypot and Its Attacks. International Journal of Computer Applications, 139(14), 14-19.

61. Banerjee, S., & Roy Chowdhury, S. (2019). A comprehensive study on honeypot: A survey. Journal of Ambient Intelligence and Humanized Computing, 10(7), 2679-2702.

62. Alazab, M., Hobbs, M., Abawajy, J., & Alazab, M. (2015). Honeypots: concepts, approaches, and challenges. Journal of Cyber Security Technology, 1(2), 89-116.

63. Miah, M. R., Batten, L., & Alam, M. S. (2016). A Survey on Blockchain Technology: Its Challenges and Opportunities. IEEE Access, 4, 1379-1394.

64. Goyal, M., & Singh, S. (2018). A Review of Blockchain Technology: Applications, Challenges and Opportunities. International Journal of Engineering and Technology, 7(4.17), 404-407.

65. Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain Technology: Beyond Bitcoin. Applied Innovation, 2(6-10), 71-81.

66. Swan, M. (2015). Blockchain: blueprint for a new economy. O'Reilly Media, Inc.

67. Zohrevandi, B., Gachpaz Hamed, S., Sargolzaei, M., & Rafeh, R. (2018). A Survey of Blockchain Technology: Architecture, Consensus, and Future Trends. In 2018 3rd International Conference on Computer and Communication Systems (ICCCS) (pp. 126-131). IEEE.

68. Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. IEEE Access, 4, 2292-2303.

69. Eyal, I., & Sirer, E. G. (2018). Majority is not enough: Bitcoin mining is vulnerable. Communications of the ACM, 61(7), 95-102.

70. Wang, S., Zhou, X., & Zhang, X. (2019). A Review on Consensus Algorithm of Blockchain. In 2019 2nd International Conference on Artificial Intelligence, Big Data and Computing (ICABDC) (pp. 239-243). IEEE.

71. Tschorsch, F., & Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. IEEE Communications Surveys & Tutorials, 18(3), 2084-2123.

72. Swan, M. (2018). Blockchain: the complete guide to understanding Blockchain technology for beginners and business professionals. Independently Published.

73. Yang, L., Hu, X., Wu, J., & Liu, Y. (2019). A distributed honeypot deployment system based on Blockchain technology. IEEE Access, 7, 35881-35890.

74. Mohamadi, A., Hosseini, S., Mousavi, S. M., & Hedayati, A. (2018). Design and implementation of a new low-interaction honeypot system using Blockchain. Future Generation Computer Systems, 81, 33-42.

75. Wazid, M., & Hasan, R. (2019). A Blockchain-based secure and robust honeypot framework for smart cities. IEEE Access, 7, 101118-101131.

76. Song, W., Hu, W., & Lu, R. (2019). A Blockchain-based decentralized honeypot system for security protection in IoT environments. Future Generation Computer Systems, 94, 207-217.

77. Zhu, B., Fu, X., Wang, Z., & Wang, H. (2018). A novel honeypot system based on Blockchain. International Journal of Security and Networks, 13(4), 221-230.

78. Li, Y., Li, Q., Li, C., & Li, Z. (2019). A Blockchain-based distributed honeypot system for intelligent manufacturing security. IEEE Access, 7, 59657-59667.

79. Jang, H. Y., Kim, J., & Kim, K. H. (2018). A Blockchain-based honeypot architecture for IoT botnet detection. In 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech) (pp. 558-565). IEEE.

80.Zhang, Y., Li, X., Li, M., & Li, W. (2019). A novel Blockchain-based collaborative honeypot system. Future Generation Computer Systems, 101, 1145-1154.

81.Wang, Q., Chen, X., Zhang, Y., & Zhao, H. (2020). A novel Blockchain-based honeypot system with a dynamic reward mechanism. International Journal of Communication Systems, 33(2), e4136.

82. Опірський І. Р., Василишин С. І., Сусукайло В. А. Аналіз загроз та безпеки технології NFC при передачі даних для автоматизованої реплікації профілю користувача // Вісник Східно-Українського національного університету імені Володимира Даля. Інформаційна безпека. – 2018. – №3/4 (31/32). – С. 37–44.

83. Опірський І. Р., Сусукайло В. А., Василишин С. І., Луковський Т. І. Розробка методу використання технології NFC для автоматизованої реплікації профілю користувача // Вісник Східно-Українського національного університету імені Володимира Даля. Інформаційна безпека. – 2018. – №3/4 (31/32). – С. 151–157.

84. І.Р. Опірський, С. Василишин, і А. Піскозуб. Аналіз використання програмних приманок як засобу забезпечення інформаційної безпеки. // Кібербезпека: освіта, наука, техніка, вип. 2, вип. 10, с. 88-97, 2020.

85. Опірський І.Р., С.І. Василишин, В.А. Сусукайло. Розслідування кіберзлочинів за допомогою приманок у хмарному середовищі. Безпека інформації, 27(1). – с.13-20. – 2021р.. https://doi.org/10.18372/2225-5036.26.15574

86. Susukailo, V., Vasylyshyn, S., Opirskyy, I., Buriachok, V., & Riabchun, O. (2021). Cybercrimes investigation via honeypots in cloud environments. Paper presented at the CEUR Workshop Proceedings, 2923 91-96.

87. Vasylyshyn, S., Opirskyy, I., Shevchenko S. (2021). Honeypot Security Efficiency versus Deception Solution. Paper presented at the CEUR Workshop Proceedings, , 3188, 229-236.

88. INFORMATION TECHNOLOGIES FOR THE SYNTHESIS OF RULE DATABASES OF AN INTELLIGENT LIGHTING CONTROL SYSTEM, Vasylyshyn, S., Lakhno, V., Alibiyeva, N., ...Pleskach, V., Lakhno, M. Journal of Theoretical and Applied Information Technologythis link is disabled, 2022, 100(5), pp. 1340–1353 (Scopus)

89. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitaliy Susukailo, Blockchain technology as the new defense of information systems and networks in cyber-spheres, NGSEC22

90. Опірський І.Р., С.І. Василишин. Перспективи військового застосування технології Blockchain, 28(2). – с.57-66. – 2022р.. https://doi.org/10.18372/2225-5036.28.16950

91. Опірський І.Р., С.І. Василишин. Розробка безпеки системи електронного урядування на основі Blockchain, 24(2). – с.58-70. – 2022р.. https://doi.org/10.18372/2410-7840.24.16931

92. Vasylyshyn, S., Susukailo, V., Opirskyy, I., Kurii, Y., Tyshyk, I. (2023). A model of decoy system based on dynamic attributes for cybercrime investigation. Eastern-European Journal of Enterprise Technologies, 1 (9 (121)), 6–20. doi: https://doi.org/10.15587/1729-4061.2023.273363 (Scopus)

93. В. Сусукайло С. Василишин, І. Опірський. «ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ ВИКОРИСТАННЯ ЧАТБОТІВ ЗІ ШТУЧНИМ ІНТЕЛЕКТОМ ДЛЯ ДОСЛІДЖЕННЯ ЖУРНАЛІВ ПОДІЙ» // НАУ: «Захист інформації». – Том 24, №4 – Київ, 2022р. – С.177-183.

94. Ivan Opirskyy, Sviatoslav Vasylyshyn, Olexsiy Vavrichen. Game theory method as optimization of Cyber Security Defence // VII Міжнародна науково-технічна

конференції “Захист інформації і безпека інформаційних систем” 30-31.05. 2019 р. – Україна, Львів, 2019р.–С.31-32.

95. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitalii Susukailo. Analysis of the use of software baits (honeypots) as a means of ensuring information security // International Workshop on Information Modeling, Zbarazh, Ukraine, 2020, 2, pp. 242–245, 9321925, DOI: 10.1109/CSIT49958.2020.9321925

96. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitalii Susukailo. Analysis of the attack vectors used by threat actors during pandemic // International Workshop on Information Modeling, Zbarazh, Ukraine, 2020, 2, pp. 261–264, 9321897, DOI: 10.1109/CSIT49958.2020.9321897

97. Опірський І. Р., Василишин С. І. АНАЛІЗ ПРОГРАМНИХ ПРИМАНОК ЯК ЗАСОБІВ МОНІТОРИНГУ ІНФОРМАЦІЇ У КІБЕРПРОСТОРІ , Збірник тез доповідей IV Всеукраїнської науково-практичної конференції молодих учених, студентів і курсантів, ст. 25.

98. Ivan Opirskyy, Sviatoslav Vasylyshyn, Upgrading network efficiency using the honeypot // VIII Міжнародна науково-технічна конференції “Захист інформації і безпека інформаційних систем” 10-11.11. 2021 р. – Україна, Львів, 2021р.–С.41-42.

99. Опірський І.Р., Василишин С.І., Стан проблеми удосконалення методології використання програмних приманок // “Технічні засоби захисту інформації”, семніар при вченій раді НАН України, Київ, Україна, 2018 р.

100. Опірський І.Р., Василишин С.І., Теорія ігор як засіб для побудови стратегії захисту з використанням програмних приманок. // “Технічні засоби захисту інформації”, семніар при вченій раді НАН України, Київ, Україна, 2020 р.

101. Опірський І.Р., Василишин С.І. Сусукайло В.А., Дослідження вразливості Zerologon // “Технічні засоби захисту інформації”, семніар при вченій раді НАН України, Київ, Україна, 2021 р.

ДОДАТОК А. Акти впровадження

N-iX

АКТ

про впровадження результатів дисертаційної роботи

Василишина Святослава Ігоровича

"РОЗРОБКА МЕТОДУ ВИКОРИСТАННЯ ПРОГРАМНИХ ПРИМАНОК ЯК ЕЛЕМЕНТІВ ЗАХИСТУ КОМП'ЮТЕРНИХ МЕРЕЖ НА ОСНОВІ ТЕХНОЛОГІЇ BLOCKCHAIN"

Комісія у складі голови – заступника директора Товариства з обмеженою відповідальністю "Н-ІКС СПЕЙС", Вітченка Віталія Миколайовича, та члена комісії – експерта з кібербезпеки, Кулініча Віталія Миколайовича, склала цей акт про те, що на основі запропонованих досліджень та розглянутої моделі динамічної системи активних пасток, а також математичного апарату обчислення динамічних атрибутів програмних приманок було розроблено та впроваджено алгоритм прогнозування несанкціонованого доступу до інформаційних мереж. Отримані результати, розроблений алгоритм та представлена на основі цього блок-схема цієї системи використані для підсилення наявних систем моніторингу та попередження несанкціонованого доступу в інформаційних мережах, що дозволило підвищити рівень захищеності та ефективності системи прогнозування про несанкціонований доступ.

Заступник директора
ТОВ "Н-ІКС СПЕЙС" _____ **Вітченко В.М.**

Експерт з кібербезпеки
ТОВ "Н-ІКС СПЕЙС" _____ **Кулініч В.М.**

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Національного університету
«Львівська політехніка»
проф. _____ Іван ДЕМИДОВ
"28" 04 2023 р.

АКТ
про використання результатів дисертаційної роботи
Василишина Святослава Ігоровича
«Розробка методу використання програмних приманок як елементів захисту комп'ютерних мереж на основі технології Blockchain» представленої на здобуття наукового ступеня доктора технічних наук за спеціальністю 125 – *Кібербезпека*

Комісія у складі – голови начальника науково-дослідної частини, д.т.н., ст. досл. Небесного Р.В. та членів: завідувача кафедри захисту інформації, д.т.н, професора Дудикевича В.Б., завідувача відділу науково-організаційного супроводу наукових досліджень, к.т.н. Лазько Г.В. і в.о. заступника начальника планово-фінансового відділу Фаст І.І., цим актом підтверджують, що результати дисертаційної роботи Василишина С.І. використовувалися при виконанні науково-дослідної роботи кафедри захисту інформації «Розроблення та удосконалення методів та засобів захисту інформації для протидії несанкціонованому доступу в інформаційно-комунікаційних мережах» (№ держреєстрації 0119U101690).

Василишином С.І. розроблено нові та вдосконалено відомі методи прогнозування та попередження про несанкціонований доступ в інформаційних мережах за рахунок використання систем програмних приманок побудованих на основі Blockchain, зокрема математичний апарат обчислення динамічних атрибутів програмних приманок, який, на відміну від відомих враховує динамічні та транс локаційні можливості блокчейн-технології Solana, модель динамічної системи активних пасток (honeypots) на основі програмних приманок, що використовують блокчейн технологію, модель динамічної системи програмних приманок, яка використовує блокчейн-технологію для підтримки безпеки, прозорості та адаптивності до зовнішніх атак, за рахунок плаваючих хостів в мережі, що дозволило підвищити ефективність захисту інформаційно-телекомунікаційних систем.

Голова комісії,
начальник науково-дослідної
частини, д.т.н. ст. досл. _____ Роман НЕБЕСНИЙ

Члени комісії:
зав. каф. захисту інформації, д.т.н. проф. _____ Валерій ДУДИКЕВИЧ

зав. відділу науково-організаційного
супроводу наукових досліджень, к.т.н. _____ Галина ЛАЗЬКО

в.о. заст. нач. планово-фінансового відділу _____ Ірина ФАСТ

ЗАТВЕРДЖУЮ

Проректор з науково-педагогічної роботи

Національного університету

«Львівська політехніка»

доц. _____ Олег ДАВИДЧАК

"___" _____ 2023 р.

АКТ

про впровадження результатів дисертаційної роботи в навчальний процес

Василишина Святослава Ігоровича

«Розробка методу використання програмних приманок як елементів захисту комп'ютерних мереж на основі технології Blockchain» представленої на здобуття наукового ступеня кандидата технічних наук за спеціальністю 125 – *Кібербезпека*

Комісія НУ «Львівська політехніка» у складі:

Голова комісії – голова науково-методичної ради інституту комп'ютерних технологій та метрології, д.т.н., проф. Байцар Р.І.

Члени комісії:

Завідувач кафедри "Захист інформації", д.т.н., проф. Дудикевич В.Б., професор кафедри "Захист інформації", к.т.н., доц. Гаранюк П.І. і доцент кафедри "Захист інформації", к.т.н., доц. Совин Я.Р.

даним актом підтверджує, що проведені дисертантом наукові дослідження виконувалися ним на кафедрі «Захист інформації» Національного університету «Львівська політехніка». Основні положення та результати дисертаційної роботи впроваджені у навчальний процес кафедри «Захист інформації» Національного університету «Львівська політехніка» при вивченні дисциплін:

- «Нормативно-правове забезпечення та міжнародні стандарти кібербезпеки» для студентів напрямку підготовки 125 "Кібербезпека", спеціалізації «Управління інформаційною безпекою», тема №3 «Загрози інформаційній безпеці. Дестабілізуючі фактори інформаційної безпеки» – аналіз стану проблеми несанкціонованого доступу в Україні; тема №4 «Методи і засоби забезпечення

інформаційної безпеки» – дослідження та аналіз моделей захисту від несанкціонованого доступу в спеціальних інформаційних мережах та аналіз існуючих підходів до протидії несанкціонованого доступу на основі теорії ігор.

Голова комісії,
голова науково-методичної ради ІКТА
д.т.н., проф. Роман БАЙЦАР

Члени комісії:
проф. каф. ЗІ, д.т.н. проф. Валерій ДУДИКЕВИЧ

доц. каф. ЗІ, к.т.н. доц. Ярослав СОВИН

доц. каф. ЗІ, к.т.н доц. Петро ГАРАНЮК

ДОДАТОК Б. Фрагменти програмних кодів моделей реалізації Blockchain мережі та контрактів

1. Block.py

```
import hashlib
import os
import json
from datetime import datetime

class BlockchainBlock:
    def __init__(self, block_dict):
        for key, value in block_dict.items():
            setattr(self, key, value)
        if not hasattr(self, 'nonce'):
            self.nonce = 'None'
        if not hasattr(self, 'hash'):
            self.hash = self.generate_hash()

    def block_header(self):
        return str(self.index) + self.previous_hash + self.data + str(self.timestamp) + str(self.nonce)

    def generate_hash(self):
        sha256 = hashlib.sha256()
        sha256.update(self.block_header().encode('utf-8'))
        return sha256.hexdigest()

    def save_block(self):
        chain_data_folder = 'chain_data'
        index_as_string = str(self.index).zfill(6)
        block_filename = f'{chain_data_folder}/{index_as_string}.json'
        with open(block_filename, 'w') as block_file:
            json.dump(self.block_info(), block_file)

    def block_info(self):
        details = {}
        details['index'] = str(self.index)
        details['timestamp'] = str(self.timestamp)
        details['previous_hash'] = str(self.previous_hash)
        details['hash'] = str(self.hash)
        details['data'] = str(self.data)
        details['nonce'] = str(self.nonce)
        return details
```

```
    def __str__(self):
        return f"BlockchainBlock<previous_hash: {self.previous_hash}, hash: {self.hash}>"

def create_genesis_block():
    initial_block_data = {
        'index': 0,
        'timestamp': datetime.now(),
        'data': 'Genesis Block',
        'previous_hash': '',
        'nonce': 0
    }
    return BlockchainBlock(initial_block_data)

if __name__ == '__main__':
    chain_data_folder = 'chain_data/'
    if not os.path.exists(chain_data_folder):
        os.mkdir(chain_data_folder)
    if not os.listdir(chain_data_folder):
        genesis_block = create_genesis_block()
        genesis_block.save_block()
```

2. mine.py

```
from block import Block
import datetime
import hashlib

NUM_LEADING_ZEROS = 5

def create_header(idx, previous_hash, content, timestamp, nonce_value):
    return str(idx) + previous_hash + content + str(timestamp) + str(nonce_value)

def compute_hash(idx, previous_hash, content, timestamp, nonce_value):
    header_str = create_header(idx, previous_hash, content, timestamp, nonce_value)
    sha256 = hashlib.sha256()
    sha256.update(header_str.encode('utf-8'))
    return sha256.hexdigest()

def perform_mining(last_block):
    idx = int(last_block.index) + 1
    timestamp = datetime.datetime.now()
    content = f"I am block #{idx}"
    previous_hash = last_block.hash
```

```
    nonce_value = 0
    block_hash = compute_hash(idx, previous_hash, content, timestamp, nonce_value)

    while str(block_hash[0:NUM_LEADING_ZEROS]) != '0' * NUM_LEADING_ZEROS:
        nonce_value += 1
        block_hash = compute_hash(idx, previous_hash, content, timestamp, nonce_value)

    mined_block_data = {
        'index': idx,
        'previous_hash': last_block.hash,
        'timestamp': timestamp,
        'data': f"Give me {idx} dollars",
        'hash': block_hash,
        'nonce': nonce_value
    }

    return Block(mined_block_data)

if __name__ == '__main__':
    node_blocks = sync.sync()
    previous_block = node_blocks[-1]
    mined_block = perform_mining(previous_block)
    mined_block.self_save()
```

3. node.py

```
from block import Block
from flask import Flask
import sync_module
import json

app = Flask(__name__)
chain_blocks = sync_module.sync()

@app.route('/Blockchain_data.json', methods=['GET'])
def get_Blockchain_data():
    chain_blocks = sync_module.sync()
    python_block_list = []

    for blk in chain_blocks:
        python_block_list.append(blk.__dict__())

    json_block_data = json.dumps(python_block_list)
    return json_block_data
```

```
if __name__ == '__main__':
    app.run()
```

4. sync.py

```
from block import Block
import os
import json

def synchronize_chain():
    synced_blocks = []
    data_directory = 'chain_data'

    if os.path.exists(data_directory):
        for file in os.listdir(data_directory):
            if file.endswith('.json'):
                file_path = f'{data_directory}/{file}'
                with open(file_path, 'r') as block_file:
                    block_data = json.load(block_file)
                    block_instance = Block(block_data)
                    synced_blocks.append(block_instance)

    return synced_blocks
```

5. contract.sol

```
pragma solidity ^0.8.3;

interface IERC20Token {
    function totalTokenSupply() external view returns (uint256);
    function accountBalance(address holder) external view returns (uint256);
    function transferTokens(address receiver, uint256 quantity) external returns (bool);
    function allowedAmount(address owner, address spender) external view returns (uint256);
    function approveSpender(address spender, uint256 quantity) external returns (bool);
    function transferFromSender(address sender, address receiver, uint256 quantity) external returns (bool);

    event TokenTransfer(address indexed from, address indexed to, uint256 value);
    event TokenApproval(address indexed owner, address indexed spender, uint256 value);
    event TransferDetails(address indexed from, address indexed to, uint256 total_Amount, uint256 reflected_amount,
uint256 total_TransferAmount, uint256 reflected_TransferAmount);
}

abstract contract ExecutionContext {
    function _sender() internal view virtual returns (address) {
```

```
        return msg.sender;
    }

    function _data() internal view virtual returns (bytes calldata) {
        this;
        return msg.data;
    }
}

library AddressUtils {
    function isContractAddress(address account) internal view returns (bool) {
        uint256 codeSize;
        assembly { codeSize := extcodesize(account) }
        return codeSize > 0;
    }

    function sendFunds(address payable recipient, uint256 amount) internal {
        require(address(this).balance >= amount, "Address: insufficient balance");
        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "Address: unable to send value, recipient may have reverted");
    }

    function executeFunction(address target, bytes memory inputData) internal returns (bytes memory) {
        return executeFunction(target, inputData, "Address: low-level call failed");
    }

    function executeFunction(address target, bytes memory inputData, string memory errorMsg) internal returns (bytes memory) {
        return executeFunctionWithValue(target, inputData, 0, errorMsg);
    }

    function executeFunctionWithValue(address target, bytes memory inputData, uint256 value) internal returns (bytes memory) {
        return executeFunctionWithValue(target, inputData, value, "Address: low-level call with value failed");
    }

    function executeFunctionWithValue(address target, bytes memory inputData, uint256 value, string memory errorMsg) internal returns (bytes memory) {
        require(address(this).balance >= value, "Address: insufficient balance for call");
        require(isContractAddress(target), "Address: call to non-contract");
        (bool success, bytes memory returnData) = target.call{ value: value }(inputData);
        return _validateCallResult(success, returnData, errorMsg);
```

```
    }

    function executeStaticFunction(address target, bytes memory inputData) internal view returns (bytes memory) {
        return executeStaticFunction(target, inputData, "Address: low-level static call failed");
    }
}     function executeStaticCall(address target, bytes memory data, string memory errorMsg) internal view returns (bytes memory) {
    require(isContractAddress(target), "Address: static call to non-contract");
    (bool success, bytes memory returndata) = target.staticcall(data);
    return _validateCallResult(success, returndata, errorMsg);
}

function executeDelegateCall(address target, bytes memory data) internal returns (bytes memory) {
    return executeDelegateCall(target, data, "Address: low-level delegate call failed");
}

function executeDelegateCall(address target, bytes memory data, string memory errorMsg) internal returns (bytes memory) {
    require(isContractAddress(target), "Address: delegate call to non-contract");
    (bool success, bytes memory returndata) = target.delegatecall(data);
    return _validateCallResult(success, returndata, errorMsg);
}

function _validateCallResult(bool success, bytes memory returndata, string memory errorMsg) private pure returns(bytes memory) {
    if (success) {
        return returndata;
    } else {
        if (returndata.length > 0) {
            assembly {
                let returndata_size := mload(returndata)
                revert(add(32, returndata), returndata_size)
            }
        } else {
            revert(errorMsg);
        }
    }
}

abstract contract Managed is ExecutionContext {
    address private _manager;
    event ManagementTransferred(address indexed previousManager, address indexed newManager);
```

```
    constructor() {
        _manager = _sender();
        emit ManagementTransferred(address(0), _manager);
    }

    function manager() public view virtual returns (address) {
        return _manager;
    }

    modifier onlyManager() {
        require(manager() == _sender(), "Managed: caller is not the manager");
        _;
    }

    function transferManagement(address newManager) public virtual onlyManager {
        require(newManager != address(0), "Managed: new manager is the zero address");
        emit ManagementTransferred(_manager, newManager);
        _manager = newManager;
    }
}

interface IUniswapV2Factory {
    event PairCreated(address indexed token0, address indexed token1, address pair, uint);

    function feeRecipient() external view returns (address);
    function feeRecipientSetter() external view returns (address);
    function getPair(address tokenA, address tokenB) external view returns (address pair);
    function allPairs(uint) external view returns (address pair);
    function allPairsLength() external view returns (uint);
    function createPair(address tokenA, address tokenB) external returns (address pair);
    function setFeeRecipient(address) external;
    function setFeeRecipientSetter(address) external;
}

interface IV2TokenPair {
    event AuthApproval(address indexed owner, address indexed spender, uint256 value);
    event AuthTransfer(address indexed src, address indexed dst, uint256 value);

    function pairName() external pure returns (string memory);
    function pairSymbol() external pure returns (string memory);
    function pairDecimals() external pure returns (uint8);
```

```
    function totalPairSupply() external view returns (uint256);
    function pairBalanceOf(address owner) external view returns (uint256);
    function pairAllowance(address owner, address spender) external view returns (uint256);
    function pairApprove(address spender, uint256 value) external returns (bool);
    function pairTransfer(address dst, uint256 value) external returns (bool);
    function pairTransferFrom(address src, address dst, uint256 value) external returns (bool);

    function DOMAIN_SEPARATOR() external view returns (bytes32);
    function PERMIT_TYPEHASH() external pure returns (bytes32);
    function nonces(address owner) external view returns (uint256);
    function permit(address owner, address spender, uint256 value, uint256 deadline, uint8 v, bytes32 r, bytes32 s) external;

    event PairMint(address indexed sender, uint256 amount0, uint256 amount1);
    event PairBurn(address indexed sender, uint256 amount0, uint256 amount1, address indexed to);
    event PairSwap(
        address indexed sender,
        uint256 amount0In,
        uint256 amount1In,
        uint256 amount0Out,
        uint256 amount1Out,
        address indexed to
    );
    event PairSync(uint112 reserve0, uint112 reserve1);

    function MINIMUM_LIQUIDITY() external pure returns (uint256);
    function factory() external view returns (address);
    function tokenA() external view returns (address);
    function tokenB() external view returns (address);
    function getReserves() external view returns (uint112 reserve0, uint112 reserve1, uint32 blockTimestampLast);
    function price0CumulativeLast() external view returns (uint256);
    function price1CumulativeLast() external view returns (uint256);
    function kLast() external view returns (uint256);
    function mintPair(address to) external returns (uint256 liquidity);
    function burnPair(address to) external returns (uint256 amount0, uint256 amount1);
    function pairSwap(uint256 amount0Out, uint256 amount1Out, address to, bytes calldata data) external;
    function pairSkim(address to) external;
    function pairSync() external;
    function initializePair(address, address) external;
}
interface IV2Router01 {
    function routerFactory() external pure returns (address);
    function WETH() external pure returns (address);
```

```
function addLiquidity(
    address tokenA,
    address tokenB,
    uint amountADesired,
    uint amountBDesired,
    uint amountAMin,
    uint amountBMin,
    address to,
    uint deadline
) external returns (uint amountA, uint amountB, uint liquidity);
function addLiquidityETH(
    address token,
    uint amountTokenDesired,
    uint amountTokenMin,
    uint amountETHMin,
    address to,
    uint deadline
) external payable returns (uint amountToken, uint amountETH, uint liquidity);
function removeLiquidity(
    address tokenA,
    address tokenB,
    uint liquidity,
    uint amountAMin,
    uint amountBMin,
    address to,
    uint deadline
) external returns (uint amountA, uint amountB);
function removeLiquidityETH(
    address token,
    uint liquidity,
    uint amountTokenMin,
    uint amountETHMin,
    address to,
    uint deadline
) external returns (uint amountToken, uint amountETH);
function removeLiquidityWithPermit(
    address tokenA,
    address tokenB,
    uint liquidity,
    uint amountAMin,
    uint amountBMin,
    address to,
```

```
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETHWithPermit(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountToken, uint amountETH);
    function swapExactTokensForTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapTokensForExactTokens(
        uint amountOut,
        uint amountInMax,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapExactETHForTokens(uint amountOutMin, address[] calldata path, address to, uint deadline)
        external
        payable
        returns (uint[] memory amounts);
    function swapTokensForExactETH(uint amountOut, uint amountInMax, address[] calldata path, address to, uint
deadline)
        external
        returns (uint[] memory amounts);
    function swapExactTokensForETH(uint amountIn, uint amountOutMin, address[] calldata path, address to, uint
deadline)
        external
        returns (uint[] memory amounts);
    function swapETHForExactTokens(uint amountOut, address[] calldata path, address to, uint deadline)
        external
        payable
        returns (uint[] memory amounts);
```

```
    function quote(uint amountA, uint reserveA, uint reserveB) external pure returns (uint amountB);
    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) external pure returns (uint amountOut);
    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) external pure returns (uint amountIn);
function getAmountsOut(uint amountIn, address[] calldata path) external view returns (uint[] memory amounts);
function getAmountsIn(uint amountOut, address[] calldata path) external view returns (uint[] memory amounts);
}
interface IV2Router02 is IV2Router01 {
    function removeLiquidityETHSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountETH);
    function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountETH);
    function swapExactTokensForTokensSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external;
    function swapExactETHForTokensSupportingFeeOnTransferTokens(
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external payable;
    function swapExactTokensForETHSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
```

```
        uint deadline
    ) external;
}
contract Test is Context, IERC20, Ownable {
    using Address for address;
    mapping (address => uint256) public _balance_reflected;
    mapping (address => uint256) public _balance_total;
    mapping (address => mapping (address => uint256)) private _allowances;
    mapping (address => bool) private _isExcludedFromFee;
    mapping (address => bool) private _isExcluded;
    mapping (address => bool) public _isBlacklisted;
    mapping (address => bool) public _isWhitelisted;
    bool public tradingOpen = false;
    bool public buyCooldownEnabled = true;
    uint8 public cooldownTimerInterval = 60;
    mapping (address => uint) private cooldownTimer;
    address[] private _excluded;
    uint256 private constant MAX = ~uint256(0);
    uint8 private   _decimals          = 9;
    uint256 private _supply_total      = 2 * 10**15 * 10**_decimals;
    uint256 private _supply_reflected  = (MAX - (MAX % _supply_total));
    string private  _name              = "Test";
    string private  _symbol            = "Test";
          uint256 public _fee_burn = 0;
    uint256 private _fee_burn_old = _fee_burn;
    address payable public _wallet_burn = payable(0x000000000000000000000000000000000000dead);
    uint256 public _fee_buyback = 0;
    uint256 private _fee_buyback_old = _fee_buyback;
    address payable public _wallet_buyback = payable(0x2445599E23C43f2C9BC001786BE436Fc8fa376eb);
    uint256 public _fee_liquidity = 0;
    uint256 private _fee_liquidity_old = _fee_liquidity;
    uint256 public _fee_denominator = 10000;
    IUniswapV2Router02 public immutable uniswapV2Router;
    address public immutable uniswapV2Pair;
    bool inSwapAndLiquify;
    bool public swapAndLiquifyEnabled = true;
    uint256 public _maxWalletToken = _supply_total;
    uint256 public _maxTxAmount = _supply_total;
    uint256 public _numTokensSellToAddToLiquidity =  ( _supply_total * 2 ) / 1000;
    uint256 public sellMultiplier = 200;
    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
    event SwapAndLiquifyEnabledUpdated(bool enabled);
```

```
    event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );
    //  PCSRouter Mainnet = 0x10ED43C718714eb63d5aA57B78B54704E256024E;
    // PCSRouter Testnet = 0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3
    address PCSRouter = 0x10ED43C718714eb63d5aA57B78B54704E256024E;
    address deadAddress = 0x000000000000000000000000000000000000dEaD;

function tokenFromReflection(uint256 reflectedAmount) public view returns(uint256) {
        require(reflectedAmount <= _supply_reflected, "Amount must be less than total reflections");
        uint256 currentRate = _getRate();
        return reflectedAmount / currentRate;
    }

    function reflectionFromToken(uint256 tokenAmount, bool deductTransferFee) public view returns(uint256) {
        require(tokenAmount <= _supply_total, "Amount must be less than total supply");
        if (!deductTransferFee) {
            uint256 currentRate = _getRate();
            return tokenAmount * currentRate;
        } else {
            uint256 fee = _calculateFee(tokenAmount);
            uint256 currentRate = _getRate();
            return (tokenAmount - fee) * currentRate;
        }
    }

    function _getRate() private view returns(uint256) {
        (uint256 reflectedSupply, uint256 totalSupply) = _getCurrentSupply();
        return reflectedSupply / totalSupply;
    }

    function _getCurrentSupply() private view returns(uint256, uint256) {
        uint256 reflectedSupply = _supply_reflected;
        uint256 totalSupply = _supply_total;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_balance_reflected[_excluded[i]] > reflectedSupply || _balance_total[_excluded[i]] > totalSupply) return
(_supply_reflected, _supply_total);
            reflectedSupply -= _balance_reflected[_excluded[i]];
            totalSupply -= _balance_total[_excluded[i]];
        }
```

```
        if (reflectedSupply < _supply_reflected / _supply_total) return (_supply_reflected, _supply_total);
        return (reflectedSupply, totalSupply);
    }

    function _calculateFee(uint256 amount) private view returns (uint256) {
        return amount * _fee_burn / _fee_denominator;
    }

    function _transfer(address sender, address recipient, uint256 amount) private {
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");
        require(amount > 0, "Transfer amount must be greater than zero");
        if(sender != owner() && recipient != owner())
            require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount");

        // Ensure the sender has enough balance
        uint256 senderBalance = _balance_reflected[sender];
        require(senderBalance >= amount, "ERC20: transfer amount exceeds balance");

        // Perform the transfer
        _balance_reflected[sender] -= amount;
        _balance_reflected[recipient] += amount;

        emit Transfer(sender, recipient, amount);
    }

    function _approve(address owner, address spender, uint256 amount) private {
        require(owner != address(0), "ERC20: approve from the zero address");
        require(spender !=address(0), "ERC20: approve to the zero address");
_allowances[owner][spender] = amount;
emit Approval(owner, spender, amount);
}function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
    _approve(_msgSender(), spender, _allowances[_msgSender()][spender] + addedValue);
    return true;
}

function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
    uint256 currentAllowance = _allowances[_msgSender()][spender];
    require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zero");
    _approve(_msgSender(), spender, currentAllowance - subtractedValue);
    return true;
}
```

```
function isExcludedFromFee(address account) public view returns(bool) {
    return _isExcludedFromFee[account];
}

function excludeFromFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = true;
}

function includeInFee(address account) public onlyOwner {
    _isExcludedFromFee[account] = false;
}

function setFee(uint256 burnFee, uint256 buybackFee, uint256 liquidityFee) external onlyOwner() {
    _fee_burn = burnFee;
    _fee_buyback = buybackFee;
    _fee_liquidity = liquidityFee;
}function setMaxTxAmount(uint256 maxTxAmount) external onlyOwner() {
    _maxTxAmount = maxTxAmount;
}function setMaxWalletToken(uint256 maxWalletToken) external onlyOwner() {
    _maxWalletToken = maxWalletToken;
}function setNumTokensSellToAddToLiquidity(uint256 _numTokens) external onlyOwner() {
    _numTokensSellToAddToLiquidity = _numTokens;
}function setSellMultiplier(uint256 multiplier) external onlyOwner() {
    sellMultiplier = multiplier;
}    function increaseAllowance(address spender, uint256 addedValue) public virtual returns (bool) {
        _approve(_msgSender(), spender, (_allowances[_msgSender()][spender] + addedValue));
        return true;
    } function decreaseAllowance(address spender, uint256 subtractedValue) public virtual returns (bool) {
        require (_allowances[_msgSender()][spender] >= subtractedValue,"ERC20: decreased allowance below zero");
        _approve(_msgSender(), spender, (_allowances[_msgSender()][spender] - subtractedValue));
        return true;
    }
    function totalFees() public view returns (uint256) {
        return _contractReflectionStored;
    }
    function isExcludedFromFee(address account) public view returns(bool) {
        return _isExcludedFromFee[account];
    }
        function ___tokenInfo () public view returns(
        uint8 Decimals,
        uint256 MaxTxAmount,
```

```
        uint256 MaxWalletToken,
        uint256 TotalSupply,
        uint256 Reflected_Supply,
        uint256 Reflection_Rate,
        bool TradingOpen,
        bool Cooldown_timer_enabled,
        uint8 Cooldown_timer_interval
        ) {
        return (_decimals, _maxTxAmount, _maxWalletToken, _supply_total, _supply_reflected, _getRate(), tradingOpen,
buyCooldownEnabled, cooldownTimerInterval  );
    }
    function ___feesInfo () public view returns(
        uint256 NumTokensSellToAddToLiquidity,
        uint256 contractTokenBalance,
        uint256 Reflection_tokens_stored
        ) {
        return (_numTokensSellToAddToLiquidity, balanceOf(address(this)), _contractReflectionStored);
    }
    function ___wallets () public view returns(
        uint256 Reflection_Fees,
        uint256 Liquidity_Fee,
        uint256 Buyback_Fee,
        uint256 Buyback_Fee_Convert_Limit,
        uint256 Buyback_Fee_Minimum_Balance,
        uint256 Marketing_Fee,
        uint256 Marketing_Fee_Convert_Limit,
        uint256 Marketing_Fee_Minimum_Balance,
        uint256 Burn_Fee,
        address Buyback_Wallet_Address,
        address Burn_Wallet_Address,
        address Marketing_Wallet_Address
        ) {
        return ( _fee_reflection, _fee_liquidity,
            _fee_buyback,_fee_buyback_convert_limit,_fee_buyback_min_bal,
            _fee_marketing,_fee_marketing_convert_limit, _fee_marketing_min_bal,
            _fee_burn,
            _wallet_buyback, _wallet_burn, _wallet_marketing);
    }
    function Change_Wallet_Marketing (address newWallet) external onlyOwner() {
        _wallet_marketing = payable(newWallet);
    }
    function Change_Wallet_Buyback (address newWallet) external onlyOwner() {
```

```
    _wallet_buyback = payable(newWallet);
}
function Change_Wallet_Burn (address newWallet) external onlyOwner() {
    _wallet_burn = payable(newWallet);
}
function deliver(uint256 tAmount) public {
    address sender = _msgSender();
    require(!_isExcluded[sender], "Excluded addresses cannot call this function");
    (uint256 rAmount,,,,,,,) = _getValues(tAmount,false);
    _balance_reflected[sender] = _balance_reflected[sender] - rAmount;
    _supply_reflected = _supply_reflected - rAmount;
    _contractReflectionStored = _contractReflectionStored + tAmount;
}
function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public view returns(uint256) {
    require(tAmount <= _supply_total, "Amount must be less than supply");
    if (!deductTransferFee) {
        (uint256 rAmount,,,,,,,) = _getValues(tAmount,false);
        return rAmount;
    } else {
        (,uint256 rTransferAmount,,,,,,) = _getValues(tAmount,false);
        return rTransferAmount;
    }
}
function tokenFromReflection(uint256 rAmount) public view returns(uint256) {
    require(rAmount <= _supply_reflected, "Amount must be less than total reflections");
    uint256 currentRate =  _getRate();
    return (rAmount / currentRate);
}
function excludeFromReward(address account) public onlyOwner() {
    require(!_isExcluded[account], "Account is already excluded");
    if(_balance_reflected[account] > 0) {
        _balance_total[account] = tokenFromReflection(_balance_reflected[account]);
    }
    _isExcluded[account] = true;
    _excluded.push(account);
}
function includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _balance_total[account] = 0;
```

```
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
function isExcludedFromReward(address account) public view returns (bool) {
    return _isExcluded[account];
}
function cooldownEnabled(bool _status, uint8 _interval) public onlyOwner {
    buyCooldownEnabled = _status;
    cooldownTimerInterval = _interval;
}
Function setNumTokensSellToAddToLiquidityt(uint256 numTokensSellToAddToLiquidity) external onlyOwner() {
    _numTokensSellToAddToLiquidity = numTokensSellToAddToLiquidity;
}
function setMaxTxPercent_base1000(uint256 maxTxPercent) external onlyOwner() {
    _maxTxAmount = (_supply_total * maxTxPercent ) / 1000;
}
 function setMaxTxTokens(uint256 maxTxTokens) external onlyOwner() {
    _maxTxAmount = maxTxTokens;
}
     function setMaxWalletPercent_base1000(uint256 maxWallPercent) external onlyOwner() {
    _maxWalletToken = (_supply_total * maxWallPercent ) / 1000;
}
 function setMaxWalletTokens(uint256 maxWallTokens) external onlyOwner() {
    _maxWalletToken = maxWallTokens;
}
    function setSwapAndLiquifyEnabled(bool _status) public onlyOwner {
    swapAndLiquifyEnabled = _status;
    emit SwapAndLiquifyEnabledUpdated(_status);
}
function s_manageExcludeFromFee(address[] calldata addresses, bool status) external onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
        _isExcludedFromFee[addresses[i]] = status;
    }}
function s_manageBlacklist(address[] calldata addresses, bool status) external onlyOwner {
    for (uint256 i; i < addresses.length; ++i) {
        _isBlacklisted[addresses[i]] = status;
    }
}
function s_manageWhitelist(address[] calldata addresses, bool status) external onlyOwner {
```

```
        for (uint256 i; i < addresses.length; ++i) {
            _isWhitelisted[addresses[i]] = status;
        }
    }
    function s_excludeFromFee(address[] calldata addresses, bool status) external onlyOwner {
        for (uint256 i; i < addresses.length; ++i) {
            _isExcludedFromFee[addresses[i]] = status;
        }
    }
    function convertLiquidityBalance(uint256 tokensToConvert) public onlyOwner {
        uint256 contractTokenBalance = balanceOf(address(this));
        if(contractTokenBalance >= _maxTxAmount) {
            contractTokenBalance = _maxTxAmount - 1;
        }
        if(tokensToConvert == 0 || tokensToConvert > contractTokenBalance){
            tokensToConvert = contractTokenBalance;
        }
        swapAndLiquify(tokensToConvert);
    }
    function purgeContractBalance() public {
        require(msg.sender == owner() || msg.sender == _wallet_marketing, "Not authorized to perform this");
         _wallet_marketing.transfer(address(this).balance);
    }
    function _getRate() private view returns(uint256) {
        (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
        return rSupply / tSupply;
    }
    function _getCurrentSupply() private view returns(uint256, uint256) {
        uint256 rSupply = _supply_reflected;
        uint256 tSupply = _supply_total;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_balance_reflected[_excluded[i]] > rSupply || _balance_total[_excluded[i]] > tSupply) return (_supply_reflected,
_supply_total);
            rSupply = rSupply - _balance_reflected[_excluded[i]];
            tSupply = tSupply - _balance_total[_excluded[i]];
        }
        if (rSupply < (_supply_reflected/_supply_total)) return (_supply_reflected, _supply_total);
        return (rSupply, tSupply);
    }
    function fees_to_bnb_manual(uint256 tokensToConvert, address payable feeWallet, uint256 minBalanceToKeep)
external onlyOwner {
        _fees_to_bnb(tokensToConvert,feeWallet,minBalanceToKeep);
```

```
    }
    function _fees_to_bnb(uint256 tokensToConvert, address payable feeWallet, uint256 minBalanceToKeep) private {
        if(tokensToConvert == 0){
            return;
        }
        if(tokensToConvert > _maxTxAmount){
            tokensToConvert = _maxTxAmount;
        }
        if((tokensToConvert+minBalanceToKeep)  <= balanceOf(feeWallet)){
            _fees_to_bnb_process(feeWallet,tokensToConvert);
        }
    }
    function _takeFee(uint256 feeAmount, address receiverWallet) private {
        uint256 reflectedReeAmount = feeAmount * _getRate();
        _balance_reflected[receiverWallet] = _balance_reflected[receiverWallet] + reflectedReeAmount;
        if(_isExcluded[receiverWallet]){
            _balance_total[receiverWallet] = _balance_total[receiverWallet] + feeAmount;
        }
        emit Transfer(msg.sender, receiverWallet, feeAmount);
    }
    function _takefees_Liquidity(uint256 amount) private {
        _takeFee(amount,address(this));
    }
    function _takefees_burn(uint256 amount) private {
        _takeFee(amount,_wallet_burn);
    }
    function _takefees_buyback(uint256 amount) private {
        _takeFee(amount,_wallet_buyback);
    }
    function _takefees_marketing(uint256 amount) private {
        _takeFee(amount,_wallet_marketing);
    }
    function _take_reflectionFee(uint256 rFee, uint256 tFee) private {
        _supply_reflected = _supply_reflected - rFee;
        _contractReflectionStored = _contractReflectionStored + tFee;
    }
    function set_sell_multiplier(uint256 Multiplier) external onlyOwner{
        sellMultiplier = Multiplier;
    }
    function set_All_Fees_Triggers(uint256 marketing_fee_convert_limit, uint256 buyback_fee_convert_limit) external
onlyOwner {
        _fee_marketing_convert_limit      = marketing_fee_convert_limit;
```

```
        _fee_buyback_convert_limit        = buyback_fee_convert_limit;
    }
    function set_All_Fees_Minimum_Balance(uint256 marketing_fee_minimum_balance, uint256 buyback_fee_minimum_balance) external onlyOwner {
        _fee_buyback_min_bal      = buyback_fee_minimum_balance;
        _fee_marketing_min_bal    = marketing_fee_minimum_balance;
    }
    function set_All_Fees(uint256 Buyback_Fee, uint256 Burn_Fees, uint256 Liquidity_Fees, uint256 Reflection_Fees, uint256 MarketingFee) external onlyOwner {
        uint256 total_fees = Burn_Fees + MarketingFee + Liquidity_Fees +  Buyback_Fee + Reflection_Fees;
        require(total_fees < 4000, "Cannot set fees this high, pancake swap will hate us!");
        _setAllFees( Burn_Fees, MarketingFee, Liquidity_Fees, Buyback_Fee, Reflection_Fees);
    }
    function removeAllFee() private {
        _fee_burn_old          = _fee_burn;
        _fee_marketing_old       = _fee_marketing;
        _fee_liquidity_old     = _fee_liquidity;
        _fee_buyback_old         = _fee_buyback;
        _fee_reflection_old    = _fee_reflection;
        _setAllFees(0,0,0,0,0);
    }
    function restoreAllFee() private {
        _setAllFees(_fee_burn_old, _fee_marketing_old, _fee_liquidity_old, _fee_buyback_old, _fee_reflection_old);
    }
    function burn_tokens_to_dead(address wallet, uint256 tokensToConvert) external onlyOwner{
        require(msg.sender == owner() || msg.sender == wallet, "Not authorized to burn");
        uint256 rTokensToConvert = tokensToConvert * _getRate();
        _balance_reflected[wallet]         = _balance_reflected[wallet]  - rTokensToConvert;
        if (_isExcluded[wallet]){
            _balance_total[wallet]         = _balance_total[wallet]      - tokensToConvert;
        }
        if (_isExcluded[deadAddress]){
            _balance_total[deadAddress]    = _balance_total[deadAddress]       + tokensToConvert;
        }
        emit Transfer(wallet, deadAddress, tokensToConvert);
    }
    function swapAndLiquify(uint256 tokensToSwap) private lockTheSwap {
        uint256 tokensHalf = tokensToSwap/2;
        uint256 contractBnbBalance = address(this).balance;
        swapTokensForEth(tokensHalf);
        uint256 bnbSwapped = address(this).balance - contractBnbBalance;
        addLiquidity(tokensHalf,bnbSwapped);
```

```
    emit SwapAndLiquify(tokensToSwap, tokensHalf, bnbSwapped);
}
function swapTokensForEth(uint256 tokenAmount) private {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0,
        path,
        address(this),
        block.timestamp);
}
function swapTokensForEthAndSend(uint256 tokenAmount, address payable receiverWallet) private {
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = uniswapV2Router.WETH();
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0,
        path,
        receiverWallet,
        block.timestamp);
}
function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
    _approve(address(this), address(uniswapV2Router), tokenAmount);
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0,
        0,
        0xe9A9F67bd63d903f1F27A4B107B17b75F9726805,
        block.timestamp);
}
function _approve(address owner, address spender, uint256 amount) private {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");
    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}
```

```
    function _transfer(address from, address to, uint256 amount) private {
        require(!_isBlacklisted[from] && !_isBlacklisted[to], "This address is blacklisted");
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");
        if (to != owner() && to != address(this) && to != address(deadAddress) && to != uniswapV2Pair && to != _wallet_marketing && to != _wallet_buyback){
            uint256 heldTokens = balanceOf(to);
            require((heldTokens + amount) <= _maxWalletToken,"Total Holding is currently limited, you can not buy that much.");}
        if(from != owner() && to != owner() && !_isWhitelisted[from] && !_isWhitelisted[to]){
            require(tradingOpen,"Trading not open yet");
        }
        if (from == uniswapV2Pair &&
            buyCooldownEnabled &&
            !_isExcludedFromFee[to] &&
            to != address(this)  &&
            to != address(deadAddress)) {
            require(cooldownTimer[to] < block.timestamp,"Please wait for cooldown between buys");
            cooldownTimer[to] = block.timestamp + cooldownTimerInterval;
        }
        if(from != owner() && to != owner()  && !_isWhitelisted[from] && !_isWhitelisted[to]){
            require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
        }
        {
            uint256 contractTokenBalance = balanceOf(address(this));
            if(contractTokenBalance >= _maxTxAmount) {
                contractTokenBalance = _maxTxAmount - 1;
            }
            bool overMinTokenBalance = contractTokenBalance >= _numTokensSellToAddToLiquidity;
            if (overMinTokenBalance &&
                !inSwapAndLiquify &&
                from != uniswapV2Pair &&
                swapAndLiquifyEnabled
            ) {
                contractTokenBalance = _numTokensSellToAddToLiquidity;
                swapAndLiquify(contractTokenBalance);
            }
            if(!inSwapAndLiquify && from != uniswapV2Pair){
                _fees_to_bnb(_fee_buyback_convert_limit,_wallet_buyback, _fee_buyback_min_bal);
                _fees_to_bnb(_fee_marketing_convert_limit,_wallet_marketing, _fee_marketing_min_bal);
            }
        }
```

```
        bool takeFee = true;
        if(_isExcludedFromFee[from] || _isExcludedFromFee[to]){
            takeFee = false;
        }
        if(!takeFee){
            removeAllFee();
        }
        (uint256 rAmount, uint256 rTransferAmount, uint256 rReflection, uint256 tTransferAmount, uint256 tBurn, uint256 tMarketing, uint256 tLiquidity, uint256 tBuyback,  uint256 tReflection) = _getValues(amount, (to == uniswapV2Pair));
        _transferStandard(from,to,amount,rAmount,tTransferAmount, rTransferAmount);
        if(!takeFee){
            restoreAllFee();
        } else{
            // functions to take all fees
            // no point to call them if there's no fees to be taken
            _takefees_burn(tBurn);
            _takefees_marketing(tMarketing);
            _takefees_Liquidity(tLiquidity);
            _takefees_buyback(tBuyback);
        }
    }
    function _transferStandard(address from, address to, uint256 tAmount, uint256 rAmount, uint256 tTransferAmount, uint256 rTransferAmount) private {
        // Update reflected Balance for sender
        _balance_reflected[from]    = _balance_reflected[from]  - rAmount;
        // Only update actual balance of sender if he's excluded from rewards
        if (_isExcluded[from]){
            _balance_total[from]    = _balance_total[from]      - tAmount;
        }
        if (_isExcluded[to]){
            _balance_total[to]      = _balance_total[to]        + tTransferAmount;
        }
        _balance_reflected[to]      = _balance_reflected[to]    + rTransferAmount;
        emit Transfer(from, to, tTransferAmount);
        emit TransferDetails(from, to, tAmount, rAmount, tTransferAmount, rTransferAmount); } receive() external payable {}
```

ДОДАТОК В. Список публікацій здобувача за темою дисертації та відомості про апробацію результатів дисертації

Статті у наукових фахових виданнях України:

1. І. Опірський, С. Василишин, А. Піскозуб. Аналіз використання програмних приманок як засобу забезпечення інформаційної безпеки. // Кібербезпека: освіта, наука, техніка, – 2020. – вип. 2, вип. 10, – С. 88-97. *Особистий внесок здобувача: представлено порівняльний аналіз програмних приманок, які існують та використовуються на комерційних та державних ринках.*
2. Опірський І.Р., Василишин С.І., Сусукайло В.А. Розслідування кіберзлочинів за допомогою приманок у хмарному середовищі. Безпека інформації, 27(1). – 2021. – С.13-20. *Особистий внесок здобувача: представлено метод використання програмних приманок та записів з журналу подій, як доказових елементів для розслідування кіберзлочинів.*
3. Опірський І.Р., С.І. Василишин. Перспективи військового застосування технології блокчейну, 28(2). – 2022р. – С.57-66. *Особистий внесок здобувача: представлено аналіз сучасного стану військових мереж комунікації та покращену модель цих мереж, які використовують технологію Blockchain.*
4. Опірський І.Р., Василишин С.І. Розробка безпеки системи електронного урядування на основі блокчейну, 24(2). – 2022р. – С.58-70. *Особистий внесок здобувача: представлено програмну архітектуру системи електронного урядування, яка використовує технологію Blockchain.*
5. В. Сусукайло С. Василишин, І. Опірський. Дослідження можливостей використання чатботів зі штучним інтелектом для дослідження журналів подій" // НАУ: "Захист інформації". – Том 24, №4 – Київ, 2022р. – С.177-183. *Особистий внесок здобувача: представлено архітектуру використання чатботів та штучного інтелекту.*
6. Опірський І.Р., Василишин С.І., Сусукайло В.А. Аналіз загроз та безпеки технології NFC при передачі даних для автоматизованої реплікації профілю користувача // Вісник Східно-Українського національного університету імені Володимира Даля. Інформаційна безпека. – 2018. – №3/4 (31/32). С. 37-44. *Особистий внесок здобувача: представлено архітектуру передачі даних у технології NFC та аналіз її захищеності.*
7. Опірський І.Р., Сусукайло В.А., Василишин С.І., Луковський Т.І. Розробка методу використання технології NFC для автоматизованої реплікації профілю користувача // Вісник Східно-Українського національного університету імені Володимира Даля. Інформаційна безпека. – 2018. – №3/4 (31/32). – С. 151–157. *Особистий внесок здобувача: представлено програмну архітектуру*

використання технології NFC для автоматизації реплікації профілю користувача.

Статті у наукових періодичних виданнях інших держав, що включені до міжнародної наукометричної бази даних (Scopus):

8. Susukailo, V., Vasylyshyn, S., Opirskyy, I., Buriachok, V., Riabchun, O. Cybercrimes investigation via honeypots in cloud environments // Paper presented at the CEUR Workshop Proceedings. 2021. Vol. 2923. P. 91-96. *(Scopus, Q4) Особистий внесок здобувача: представлено архітектуру використання програмних приманок в хмарних середовищах.*
9. Vasylyshyn, S., Opirskyy, I., Shevchenko S. (2021). Honeypot Security Efficiency versus Deception Solution // Paper presented at the CEUR Workshop Proceedings. 2021. Vol. 3188. P. 229-236. *(Scopus, Q4) Особистий внесок здобувача: представлено аналіз рівня захищеності за захисних механізмів технологій приманок та обману.*
10. Vasylyshyn, S., Lakhno, V., Alibiyeva, N., Alibiyeva, Z., Sauanova, K., Pleskach, V., Lakhno, M. Information technologies for the synthesis of rule databases of an intelligent lighting control system // M. Journal of Theoretical and Applied Information Technologythis link is disabled. 2022. Vol. 100(5). P. 1340-1353 *(Scopus, Q3) Особистий внесок здобувача: представлено метод використання баз даних та збору інформації в "розумних"мережах.*
11. Vasylyshyn, S., Susukailo, V., Opirskyy, I., Kurii, Y., Tyshyk, I. A model of decoy system based on dynamic attributes for cybercrime investigation // Eastern-European Journal of Enterprise Technologies. 2023. Vol. 1 (9 (121)). P. 6-20. *(Scopus, Q3) Особистий внесок здобувача: представлено метод використання програмних приманок побудованих з використанням динамічних атрибутів технології Blockchain.*

Наукові публікації у збірниках матеріалів та тез конференцій:

12. Ivan Opirskyy, Sviatoslav Vasylyshyn, Olexsiy Vavrichen. Game theory method as optimization of Cyber Security Defence // VII Міжнародна науково-технічна конференції "Захист інформації і безпека інформаційних систем". Україна, Львів, 30-31 травня 2019. С.31-32. *Особистий внесок здобувача: представлено математичну модель, яка використовує метод теорії ігор для оптимізації криптографічних обчислень.*
13. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitalii Susukailo. Analysis of the use of software baits (honeypots) as a means of ensuring information security // International Workshop on Information Modeling. Zbarazh, Ukraine, 2020. Vol. 2, P. 242–245, 9321925. *(Scopus, QX) Особистий внесок здобувача: представлено*

порівняльну характеристику сучасних програмних приманок, проведено порівняння найпоширеніших програмних маркерів за допомогою формалізованих критеріїв.

14. Sviatoslav Vasylyshyn, Ivan Opirskyy, Vitalii Susukailo. Analysis of the attack vectors used by threat actors during pandemic // International Workshop on Information Modeling. Zbarazh, Ukraine, 2020. Vol. 2, P. 261–264, 9321897. *(Scopus, QX) Особистий внесок здобувача: представлено аналіз сучасних векторів атак зловмисників на мережеві системи.*
15. Опірський І.Р., Василишин С.І. Аналіз програмних приманок як засобів моніторингу інформації у кіберпросторі // Збірник тез доповідей IV Всеукраїнської науково-практичної конференції молодих учених, студентів і курсантів. – С. 25. *Особистий внесок здобувача: представлено аналіз програмних приманок, як засобу захисту мережевої інфраструктури у кіберпросторі.*
16. Ivan Opirskyy, Sviatoslav Vasylyshyn, Upgrading network efficiency using the honeypot // VIII Міжнародна науково-технічна конференції "Захист інформації і безпека інформаційних систем". Україна, Львів, 10-11 листопада 2021. С.41-42. *Особистий внесок здобувача: представлено модель використання програмних приманок, як засобів для покращення якості використання комп'ютерних мереж.*
17. Опірський І.Р., Василишин С.І., Стан проблеми удосконалення методології використання програмних приманок // "Технічні засоби захисту інформації", семінар при науковій раді НАН України. Київ, Україна, 2018. *Особистий внесок здобувача: представлено аналіз методології використання програмних приманок та виділення шляхів її удосконалення.*
18. Опірський І.Р., Василишин С.І., Теорія ігор як засіб для побудови стратегії захисту з використанням програмних приманок. // "Технічні засоби захисту інформації", семніар при науковій раді НАН України. Київ, Україна, 2020. *Особистий внесок здобувача: представлено модель аналізу захищеності комп'ютерних мереж з використанням програмних приманок та принципів теорії ігор.*
19. Опірський І.Р., Василишин С.І. Сусукайло В.А., Дослідження вразливості Zerologon // "Технічні засоби захисту інформації", семінар при науковій раді НАН України, Київ, Україна. 2021. *Особистий внесок здобувача: представлено аналіз вразливостей технології Zerologon.*