

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
« ____ » _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи автоматизованого
управління системами типу Honeypot”

Виконав здобувач вищої освіти
IV курсу, групи КІ-23-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Косяченко І.В.
« ____ » _____ 2026 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
« ____ » _____ 2026 р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Освітній ступінь бакалавр
Галузь знань 12 "Інформаційні технології"
Спеціальність 123 "Комп'ютерна інженерія"
Освітньо-професійна (освітньо-наукова) програма "Комп'ютерна інженерія"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 27 » лютого 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Косяченку Іллі Вадимовичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи автоматизованого управління системами типу Honeypot*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 126-02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту *23.05.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою роботи є розробка програмного забезпечення системи автоматизованого управління системами типу Honeypot*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи *1 аркуш*

Функціональна схема системи *1 аркуш*

Діаграма процесів *1 аркуш*

Блок-схема алгоритму роботи додатку *2 аркуша*

7. Дата видачі завдання « 27 » лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 27 » лютого 2026 р.

Підпис керівника

Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 27 » лютого 2026 р.

Підпис здобувача

Косяченко І.В.
(прізвище та ініціали)

АНОТАЦІЯ

Косяченко І.В. Програмне забезпечення системи автоматизованого управління системами типу Honeypot. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи автоматизованого управління системами типу Honeypot.

Метою розробки є програмне забезпечення системи автоматизованого управління системами типу Honeypot.

Результат роботи – програмна реалізація системи автоматизованого управління системами типу Honeypot.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, автоматизоване управління системами, Honeypot

ABSTRACT

Kosiachenko I.V. Software for the Honeypot type automated control system. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the Honeypot type automated control system.

The purpose of the development is software for the Honeypot type automated control system.

The result of the work is the software implementation of the Honeypot type automated control system.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with OS Windows 10/11.

The program was developed in the Python environment.

Keywords: computer engineering, automated systems management, Honeypot

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ	2
ВСТУП.....	3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ	5
1.1 Призначення системи.....	5
1.2 Область застосування.....	6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	13
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.....	13
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування.....	17
2.3 Розгорнута постановка завдання	20
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	21
3.1 Опис функціонування системи	21
3.2 Розробка структурної схеми.....	29
3.3 Розробка функціональної схеми	32
3.4 Розробка діаграми процесів.....	39
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ.....	41
4.1 Розробка блок-схем та опис алгоритмів функціонування системи.....	41
4.2 Захист розробленого програмного забезпечення.....	57
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	59
6 ОСНОВНІ ВИСНОВКИ.....	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	63

					ВКРБ-123.26.0032.00.00.ПЗ			
Вим.	Арк.	№ докум.	Підп.	Дата	Програмне забезпечення системи автоматизованого управління системами типу Honeypot	Літ.	Аркуш	Аркушів
Розроб.	Косяченко І.В.					Б	1	69
Перев.	Смірнов О.А.							
Н.контр.	Коваленко А.С.					ЦНТУ КІ-23-МБ		
Затв.	Смірнов О.А.							

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ДПФ	—	дискретне перетворення Фур'є
ДПХ	—	дискретне перетворення Хартлі
ЕОМ	—	електрона обчислювальна машина
НСД	—	несанкціонований доступ
ПЗ	—	програмне забезпечення
СУБД	—	система управління базами даних
СВВ	—	система виявлення вторгнень
ШПФ	—	швидке перетворення Фур'є
ШПХ	—	швидке перетворення Хартлі

К6ПЗ-2026

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		2

ВСТУП

Актуальність теми. «Honeypot» – це механізм безпеки, який створює віртуальну пастку для заманювання зловмисників. Навмисно скомпрометована комп'ютерна система дозволяє зловмисникам використовувати вразливості, щоб ви могли вивчити їх для покращення своїх політик безпеки. Ви можете застосувати «приманку» до будь-якого обчислювального ресурсу, від програмного забезпечення та мереж до файлових серверів та маршрутизаторів.

«Honeypot» – це тип технології обману, яка дозволяє зрозуміти моделі поведінки зловмисників. Команди безпеки можуть використовувати «ханіпоти» для розслідування порушень кібербезпеки та збору інформації про те, як діють кіберзлочинці. Вони також зменшують ризик хибнопозитивних результатів порівняно з традиційними заходами кібербезпеки, оскільки навряд чи привернуть законну діяльність.

Honeypots відрізняються залежно від дизайну та моделей розгортання, але всі вони є приманками, призначеними для вигляду легітимних, вразливих систем, щоб привабити кіберзлочинців.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи автоматизованого управління системами типу Honeypot.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

- Огляд існуючих систем автоматизованого управління системами типу Honeypot.
- Дослідження системи автоматизованого управління системами типу Honeypot.
- Програмна реалізація системи автоматизованого управління системами типу Honeypot.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		3

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі автоматизованого управління системами типу Нoneурот.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи автоматизованого управління системами типу Нoneурот, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

К6ПЗ_2026

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Існує два основних типи конструкцій Honeypot:

- Виробничі приманки – служать системами-приманками всередині повністю працюючих мереж і серверів, часто як частина системи виявлення вторгнень (IDS). Вони відволікають увагу злочинців від реальної системи, одночасно аналізуючи шкідливу активність, щоб допомогти зменшити вразливості.

- Дослідницькі приманки – використовуються для освітніх цілей та підвищення безпеки. Вони містять відстежувані дані, які можна відстежити у разі крадіжки для аналізу атаки.

Типи розгортань Honeypot

Існує три типи розгортання honeypot, які дозволяють зловмисникам виконувати різні рівні шкідливої діяльності:

- Чисті приманки (honeypots) – повноцінні виробничі системи, які відстежують атаки за допомогою виявлення помилок у каналі, що з'єднує приманку з мережею. Вони прості.

- Маловзаємодіючі приманки – імітують сервіси та системи, які часто привертають увагу злочинців. Вони пропонують метод збору даних від сліпих атак, таких як ботнети та шкідливі програми-черв'яки.

- Приманки з високою взаємодією – складні налаштування, що поведуться як реальна виробнича інфраструктура. Вони не обмежують рівень активності кіберзлочинця, надаючи розширені знання з кібербезпеки. Однак вони потребують вищого обслуговування та потребують експертних знань і використання додаткових технологій, таких як віртуальні машини, щоб гарантувати, що зловмисники не зможуть отримати доступ до реальної системи.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

Обмеження Honeypot

Безпека honeypot має свої обмеження, оскільки honeypot не може виявляти порушення безпеки в легітимних системах і не завжди ідентифікує зловмисника. Існує також ризик того, що, успішно використавши honeypot, зловмисник може перейти до іншої мережі, щоб проникнути в реальну виробничу мережу. Щоб запобігти цьому, потрібно забезпечити належну ізоляцію honeypot.

Щоб масштабувати свої операції з безпеки, ви можете поєднувати «honeypots» з іншими методами. Наприклад, стратегія «канарейкової пастки» допомагає знаходити витoki інформації, вибірково обмінюючись різними версіями конфіденційної інформації з підозрюваними шахраями або інформаторами.

1.2 Область застосування

Використання виробничих Honeypot

Honeypot можуть бути використані для основних завдань забезпечення інформаційної безпеки [3]:

- попередження;
- виявлення;
- протоколювання.

Попередження

У термінах безпеки, попередження означає здійснення функцій, що не дозволяють виявити який-небудь деструктивний вплив зловмисників на систему. Існує безліч інструментів для попередження несанкціонованої активності, наприклад, міжмережеві екрани, які контролюють увесь вхідний і вихідний з мережі трафік, або системи автентифікації, такі як стійкі паролі, цифрові сертифікати. Таким чином, існує можливість визначити, хто авторизований для доступу до ресурсів. Такі механізми, як шифрування, застерігають зловмисників від доступу до критичної інформації, наприклад,

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

паролям або конфіденційним документам. Яку ж роль тут відіграють Honeypot? Яким чином Honeypot стримують зловмисників? Якщо Honeypot некоректно введений в експлуатацію, то він може створити ризик, надавши зловмисникові вікно в організацію. При цьому хитрість полягає в тому, що зловмисникові прийде витрачати свій час і ресурси на злом Honeypot, тоді як справжня система буде перебувати в безпеці.

Інша концепція ґрунтується на тому, що зловмисників віджахне встановлений в організації Honeypot. У кожному разі зловмисник може вирішити не витрачати час і ресурси на злом Honeypot. Обидві концепції є психологічною зброєю для заплутування зловмисника.

У той час як обман і стримування можуть запобігти нападам на виробничі системи, більшість організацій порохують більш раціональним використовувати свій час і ресурси для забезпечення й поліпшення безпеки систем. Адже який зміст буде в розгортанні Honeypot, щоб обдурити, що нападав, якщо системи організації усе ще використовують уразливі сервіси, додатки, до яких потрібно застосувати патчи, а персонал використовує паролі, які є досить простими? Обман і стримування можуть внести свій внесок у запобігання багатьох наслідків, викликаних діяльністю зловмисника, але не варто вважати даний підхід панацеєю.

Обман і стримування також не в змозі запобігти самим звичайним з нападам. Більшість зловмисників зосереджена на нападі на велику кількість систем. Вони роблять це з використанням підготовлених автоматизованих засобів. Ці зловмисники не витрачають час, аналізуючи цільові системи. Вони просто намагаються атакувати, вражаючи так багато комп'ютерів, скільки можливо, і дивляться, що відбувається. Для обману або стримування зловмисник повинен не квапитися, щоб увести “погану” інформацію, щоб Honeypot проаналізував її. Більшість зловмисників сьогодні не витрачає час на аналіз своїх цілей. Вони просто нападають на систему й потім рухаються далі. Обман і стримування спроектовані як психологічна зброя, щоб збентежити людей. У

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

цьому випадку дана зброя зазнає невдачі. Ще гірше, що більшість нападів навіть не здійснюється людьми. Вони звичайно виконуються автоматизованими засобами, наприклад, з використанням мережних хробаків. Обман або стримування не зможуть запобігти нападам такого типу.

Обман і стримування можуть працювати для організацій, які прагнуть захистити ресурси високого значення. У цих випадках існує зловмисник, що аналізує інформацію, яка йде від Honeypot. Ціль полягає в тому, щоб збентежити кваліфікованих зловмисників. На відміну від зловмисників, які зосереджують на простому, ці атакуючі ретельно вибирають свої цілі й аналізують одержувану інформацію.

У такому випадку, засіб Honeypot може використовуватися, щоб обдурити або удержати зловмисника. При цьому час і ресурси, залучені в розгортання Honeypot для того, щоб запобігати нападам, особливо при запобіганні, заснованому на обмані або стримуванні, краще використовувати для поліпшення наявної безпеки системи, тому що Honeypot не зможе запобігти наявним уразливостям у системі.

Виявлення

Друге завдання – виявлення; процес виявлення й оповіщення про несанкціоновану діяльність. Рано або пізно, попередження зазнає невдачі, і зловмисник зможе проникнути в систему. Існує кілька причин, чому дане подія може трапитися: у правилах міжмережевого екрана може бути виявлена помилка, співробітник міг використовувати простий пароль, або ж у додатку виявлена нова уразливість. Є достатня кількість методів, щоб проникнути в систему. Попередження може тільки зм'якшити ризик, але ніколи не зможе його усунути. Для виконання розглянутої функції вже існують такі технології, як мережні системи виявлення вторгнень – застосунок, спроектований для контролю мережі й виявлення будь-якої зловмисної діяльності. Є також програми, призначені для контролю системних файлів реєстрації і їх аналізу на зловмисні події. Ці застосунки не запобігають нападам зловмисників, але вони мають можливість

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		8

повідомити, якщо дані напади успішні. Яким чином засоби Honeypot допомагають виявляти зловмисну або підозрілу діяльність? Тим часом, як Honeypot не вносять достатнє значення в попередження, вони відіграють більшу роль у виявленні.

Процес виявлення в принципі відносно трудомісткий, тому що існують такі поняття, як неправильні спрацьовування, пропуски й агрегація даних. Неправильні спрацьовування – це, коли система неправильно виводить попередження про підозрілу або злонаміреної активності. У цьому випадку нормальний трафік може бути прийнятий за спробу нападу або підозрілу діяльність. Пропуски – повна протилежність: система не в змозі виявити напад. Агрегація даних – підсумовування всіх даних, необхідних для виявлення, для наступного аналізу. Однократне неправильне спрацьовування – не проблема. Проблеми починаються, коли неправильні спрацьовування трапляються сотні або навіть тисячі раз у день. Системні адміністратори можуть одержати дуже багато попереджень протягом одного дня, таким чином, обробити всі попередження не буде представлятися можливим. Дані системи стають неефективними, тому що адміністратори, у більшості випадків, перестають обертати на них увага. Точно також системні адміністратори можуть одержувати зовсім невелику кількість попереджень – і це може говорити про те, що в системі достатнє число пропусків. Проблема агрегації даних пов'язана з тим, що даних – досить багато: реєстраційні журнали системи, журнали застосунків – усі ці дані необхідно агрегувати таким чином, щоб надалі використовувати для виявлення.

Завдяки своїй простоті, засоби Honeypot ефективно співвідносяться з даними трьома аспектами. Більшість засобів Honeypot не має ніякого промислового трафіка, таким чином, імовірність неправильних спрацьовувань мінімальна. Єдина можливість, коли може відбутися неправильне спрацьовування, це коли відбулася помилка, наприклад, яка-небудь помилка Dns-сервера або помилковий запит на сервер Honeypot. В усі інших випадках Honeypot генерує коректні попередження.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		9

Noneurort також не має проблем із пропусками атак. Фактично, одна з первинних вигід – вони можуть виявити новий напад насамперед на підставі дій системної активності, а не сигнатур. Noneurort може використовувати базу сигнатур для однозначного визначення типу атаки, якщо ставиться таке завдання. Але вся взаємодія з Noneurort вважається неправомірним, тому нові типи атак виявляються точно так само, як і раніше відомі. Простота Noneurort також вирішує й третю проблему: агрегацію даних. Noneurort створюють досить невелика кількість даних, тому що коректний трафік не враховується. Немає ніякого правильного промислового трафіка, який буде зареєстрований. Noneurort генерують тільки декілька мегабайт даних у день, більшість яких має високе значення. Це робить надзвичайно простим діагностувати корисну інформацію від Noneurort.

Варто зробити вивід, що Noneurort успішно допомагають виконувати в здійсненні функцій виявлення, але ніяк не можуть виступати єдиним рішенням. Як і скрізь – тут потрібно пам'ятати про головний аспект інформаційної безпеки – безпека повинна здійснюватися комплексно.

Протоколювання

Раніше згадувалася проблема агрегації зібраних даних. Із цією проблемою незаперечно зв'язане завдання подальшого розбору інформації з метою вивчення дій зловмисника. Як правило, більшість засобів забезпечення безпеки генерує досить об'ємні протоколи. Завдання подальшого виділення необхідної для аналізу інформації може виявитися скрутною.

Крім того, існує потенційна можливість взагалі втратити інформацію, якщо комп'ютер, що атакується, буде ушкоджений зловмисником. Засіб Noneurort має кілька передумов для вирішення подібних проблем.

По-перше, як уже неодноразово згадувалося, Noneurort збирає тільки необхідну інформацію, тому завдання виділення необхідної інформації зводиться до мінімальної по часових витратах.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		10

По-друге, засіб Honeypot має можливість протоколювати інформацію й відправляти протоколи на сторонній сервер, що запобігає можливій втраті даних у випадку злому.

Коли атака була успішно виявлена, потрібно деяким чином зреагувати на дану подію. Після злому необхідно визначити, чи залишив зловмисник які-небудь чорні ходи або логічні бомби, модифікував або вилучив доступну інформацію, наприклад облікові записи користувачів. Навіть якщо зловмисник застосував методи приховання своїх дій, такі як зміни файлів системної реєстрації, ці дії однаково можуть бути простежені. Наприклад, існує можливість визначити кожний крок зловмисника, проаналізувавши атрибути файлів. У більшості операційних систем є дані по кожному файлу, коли до нього був здійснений доступ, або він був змінений. Є автоматизовані засоби, засновані на аналізі параметрів файлів, завдяки яким можна відтворити послідовність дій у системі. Але тут має місце наступна проблема. При великій активності в системі й мережі різна правомірна діяльність відбувається постійно: запускаються процеси, використовуються й змінюються файли, заповнюється й звільняється пам'ять. Імовірність того, що дії зловмисника будуть загублені або переписані коректними подіями системи – досить велика. У результаті навіть із використанням наявних автоматизованих засобів визначити дії зловмисника за ушкодженим даними буде вкрай важко.

Ще одна проблема полягає в тому, що після виявлення атаки в більшості випадків система повинна бути зупинена для аналізу. Однак у більшості організацій припинити роботу реальної системи не представляється можливим. На зламаному комп'ютері під захистом міжмережевого екрана й системи виявлення вторгнень може перебувати критичний сервер, який використовується всією організацією. Останов такого сервера може привести до більших втрат. Засіб Honeypot вирішує й ці проблеми.

Як говорилося раніше, Honeypot не обробляє ніякого правомірного трафіку, увесь трафік – підозрілий. Honeypot також не використовується для

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		11

внутрішнорганізаційних потреб. Саме тому всі дії зловмисника, крім того, що постійно записуються, також можуть бути відновлені раніше наведеним способом, тому що дані для відновлення не будуть загублені, ушкоджені або переписані. Аналогічно, останок сервера Honeypot ніяк не вплине на роботу організації, тому що даний сервер не містить ніякої критичної інформації.

Таким чином, організації можуть сфокусуватися на аналізі отриманих від Honeypot даних і застосувати згодом результати до реальних систем. Honeypot – не є єдиним рішенням для здійснення подібної реакції, це тільки інструмент, який допомагає в даному питанні. Перший застосунок, який повинний бути прийняте до інциденту, – створення резервних копій критичних файлів, відбудовних дискових образів або підготовка, що відновлюють програмних засобів. Honeypot, у свою чергу, розкриває, яким чином і де саме здійснював свою діяльність зловмисник. Усі розглянуті функції, на додаток до здійснення яких може бути використаний засіб Honeypot, як правило, необхідні організаціям, мета яких – підвищити свій рівень безпеки. Існує ж можливість використання Honeypot з метою дослідження поведінки й засобів зловмисників.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи автоматизованого управління системами типу Honeypot, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Back Officer Friendly

Back Officer Friendly (надалі, BOF) – один з найпростіших Honeypot.

На рисунку 2.1 наведене головне вікно програми BOF, запущеної на імітацію всіх сервісів.

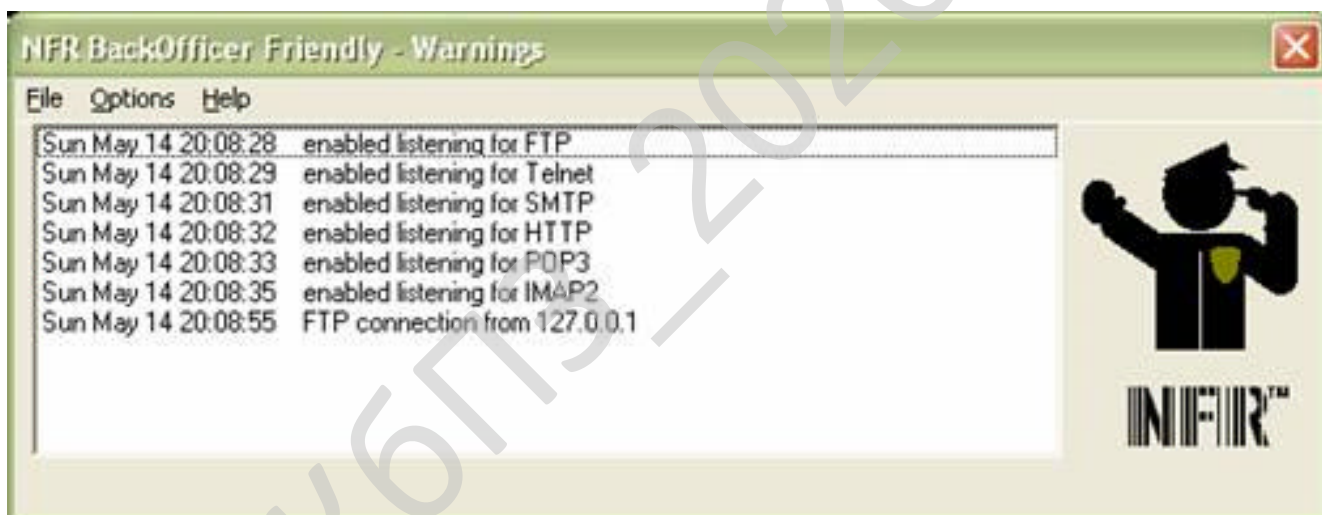


Рисунок 2.1 – Головне вікно програми Back Officer Friendly

BOF не здійснює детальну імітацію сервісів. Як тільки відбувається з'єднання, BOF виводить повідомлення про неприступність сервісу й через невеликий таймаут робить розрив з'єднання. Ціль BOF – моніторинг подій і збереження протоколу взаємодії [2].

Advanced Port Scanner Honeypot

Advanced Port Scanner (надалі, APS) – також є Honeypot слабкої взаємодії, може бути запущений на платформі Windows.

Даний Honeypot може бути використаний з метою:

- виявлення факту сканування портів (TCP, UDP) і розсилання UDP broadcast пакетів для заданих портів. Список портів, що прослуховуються, заданий у базі даних формату XML. При встановленні з'єднання програма може імітувати відповідь сервера (передані в якості відповіді дані зберігаються в базі даних налаштувань). Програма містить фільтр, що налаштовується, який дозволяє ігнорувати спроби сканування портів із заданих хостів або мереж;
- ведення протоколювання спроб сканування портів (текстовий протокол);
- передачі оповіщення адміністраторам за допомогою електронної пошти або NET SEND. Адміністратор може набудувати періодичність і состав переданої в листі інформації;
- передачі даних про спроби сканування портів службі Syslog на заданих серверах;
- ведення деталізованої статистики процесу сканування портів для кожного хоста й порту. статистика може проглядатися в реальному часі або експортуватися в текстовий файл;
- виявлення спроб атак Dos (відмова в обслуговуванні) на порти, що прослуховуються;
- блокування роботи описаних у базі налаштувань троянських програм (використання портів, що прослуховуються програмою, іншими додатками неможливо).

Під час роботи програма створює іконку в tray, при виявленні сканування портів включається анімація іконки, видається звукові сигнали, можливий автоматичний вивід на екран головного вікна програми.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

Kfsensor

Honeyrot Kfsensor розроблений компанією Keyfocus. Даний Honeyrot, як і раніше розглянуті, дозволяє виявляти зловмисників і мережних хробаків, імітуючи вразливі сервіси системи. Kfsensor спроектований для використання на Windows-системах і містить деякі інноваційні й унікальні можливості, такі як:

- вилучене адміністрування;
- сумісність із системою виявлення вторгнень Snort;
- емуляція мережних Windows-протоколів.

Kfsensor має досить простий, але при цьому із широкими можливостями й деталізацією інтерфейс керування. Головне вікно програми Kfsensor презентовано на рисунку 2.2.

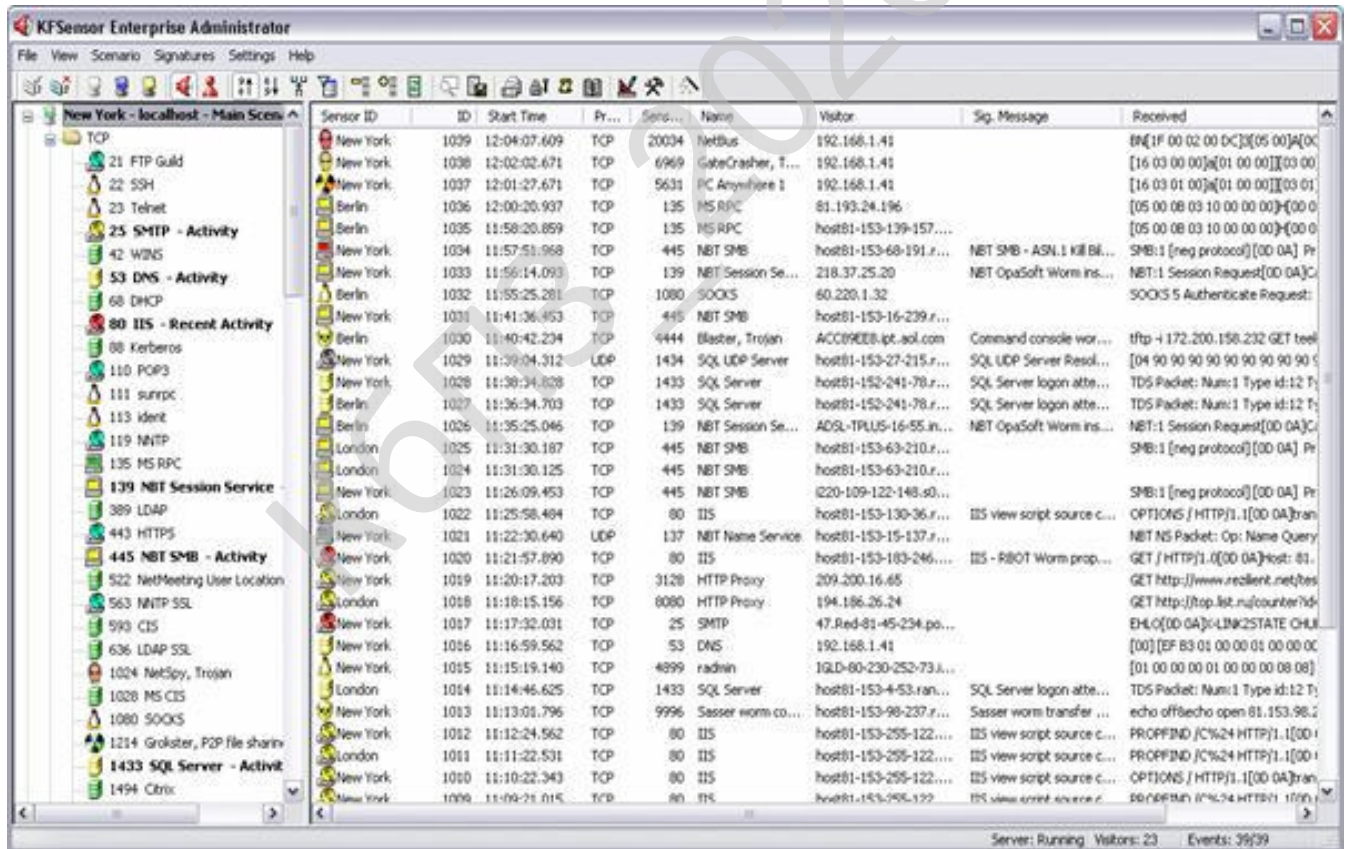


Рисунок 2.2 – Головне вікно програми Kfsensor

Kfsensor містить базу даних сигнатур, таким чином, відбувається не тільки виявлення зловмисника й запис його дій. Існує можливість надалі побачити, яку саме уразливість намагався використувувати даний зловмисник для проникнення в систему. Сигнатури Kfsensor засновані на складанні правил атаки, аналогічних правилам системи виявлення Snort. Дані правила можуть бути імпортовані з Snort в Kfsensor.

Kfsensor містить емуляцію таких мережних протоколів Windows як Netbios / SMB / CIFS. Ця унікальна здатність дозволяє виявляти велика кількість атак на файли відкритого доступу й адміністративні сервіси Windows.

Ще одна незаперечна перевага Honeypot Kfsensor – можливість розширення. Існує можливість написання власних скриптів відповідей і поведінки, збільшення бази даних сигнатур.

Одна з інноваційних можливостей – вилучене адміністрування Honeypot. Даний засіб Honeypot можна настроїти й звістки повноцінне керування віддалено. Для забезпечення безпеки, взаємодія з вилученої Honeypot-системою проводиться за допомогою шифрування й автентифікації.

Завдяки подібній архітектури, а також таким розглянутим можливостям, як емуляція мережних протоколів, даний Honeypot можна віднести до виду Honeypot середньої взаємодії [2].

Mantrap

Mantrap – комерційний засіб Honeypot сильної взаємодії, створене компанією Recourse Technologies.

Mantrap має деякі обмеження. По-перше, у силу того, що Mantrap не імітує системи, а використовує технологію “пасток”, те основа всіх операційних систем – одна. Також Mantrap підтримує лише деякі операційні системи. При цьому сам Honeypot може функціонувати тільки на комп'ютері з операційною системою Solaris. Крім того, Mantrap використовує особливу установку Solaris з необхідними параметрами, тому поки не існує можливості встановити даний Honeypot на комп'ютери із системами NT, BSD або Linux. Схожий застосунок

надає продукт Virtual Machine Ware для Windows-платформ. VMWare дозволяє встановлювати повноцінні операційні системи на один комп'ютер, однак, тому що VMWare не є Honeyrot, те не містить за замовчуванням ніяких засобів записи дій зловмисника.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність. Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту. У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка. Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об'єкта можна запитати під час виконання програми. Для кожного з об'єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		17

- у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;
- гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;
- розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;
- Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;
- у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив'язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.
2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).
3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.
4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.
5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.
6. Тестування (автоматизація цього процесу).
7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проекту.

Серед найпопулярніших фреймворків для Python:

- Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;
- Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;
- Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускню кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи автоматизованого управління системами типу Honeyrot.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформулювати висновки про виконаний обсяг робіт та одержані результати.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		20

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Якщо необхідно вивчити мотивації поведінки зловмисників, методи їх атак і засобів – тоді потрібно побудувати складний Honeypot, який надає зловмисникові повноцінну операційну систему, з якої він буде взаємодіяти, і забезпечити високий рівень протоколювання.

Ознаки, використовувані при класифікації

Процес установки й налаштування – характеризує час і зусилля при установці й налаштуванню Honeypot. Таким чином, чому складніше Honeypot і чим більш широке завдання, які перед ним ставляться – тем значніше даний параметр. Функціональність, що збільшується, яка надається зловмисникові, вимагає установки й налаштування більшої кількості сервісів і здатності обробки більшого числа команд. Простий процес установки не містить у собі вимог по конфігуруванню або завданні додаткових параметрів роботи засоби Honeypot, середній – вимагає установки або вибору обмеженого числа параметрів (наприклад, вибір варіанта імітуємого сервісу) для коректної роботи засобу, складний – необхідне досить детальне налаштування (звичайно сполучена з установкою), завдання великої кількості додаткових параметрів функціонування засоби Honeypot.

Процес використання й підтримки – характеризує час і зусилля при використанні й підтримці Honeypot після процесу установки й налаштування. Очевидним образом впливає, що чим вище функціональність Honeypot, тим складніше його використання, і тим більше часу й зусиль вимагає його підтримка. Простий процес використання й підтримки має на увазі мінімальна кількість часу, необхідне для підтримки засобу, а також простоту у використанні (наприклад, для коректного використання необхідно лише запустити програму).

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		21

Середній рівень додає в процес використання додаткові дії, спрямовані на налаштування Noneuror у процесі функціонування, необхідність участі адміністратора для коректування поведінки й т.п. Складний рівень містить у собі необхідні дії у використанні й наступній підтримці засобу (наприклад, відновлення програмного забезпечення, відновлення оточення для подальшого дослідження після того, як була зроблена взаємодія).

Збір даних – характеризує обсяг даних, який Noneuror може зібрати про зловмисника і його активності. Якщо Noneuror є спеціально підготовленою операційною системою, то даних про зловмисника і його діях може бути зібране досить багато. Якщо ж Noneuror – імітація якого-небудь системного сервісу, то кількість даних скорочується. Останній варіант являє приклад простого збору даних. Змінним збором даним характеризуються засоби Noneuror, де існують додаткові налаштування на вибір подій, що збираються. Розширений збір даних має на увазі засіб Noneuror сильної взаємодії.

Рівень протоколювання – характеризує ступінь деталізації, з якого буде зроблене протоколювання. Чим вище рівень протоколювання, тем більшу деталізацію мають записи протоколу програми. Низький рівень протоколювання визначається невеликим рівнем подробиці зібраних даних. Як правило, даним рівнем мають засоби Noneuror слабкої взаємодії, коли протоколюються тільки IP-адреса джерела взаємодії й дані, що виходять від зловмисника. Середній рівень протоколювання може містити в собі протокол обох сторін взаємодії, а також додаткові дані (наприклад, конкретний час приходу даних, ідентифікатори взаємодій і т.п.). Високим рівнем протоколювання мають Noneuror сильної взаємодії, коли засіб бере на себе обов'язок протоколювати всі події, що відбуваються в системі при взаємодії.

Рівень імітації – характеризує ступінь імітації сервісу. Простий рівень імітації має на увазі практично повна відсутність підтримки функціональності імітуємого сервісу (наприклад, можливість виводити тільки привітальне повідомлення при з'єднанні). Середній рівень має на увазі досить докладну

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		22

імітацію сервісу, з урахуванням особливостей його роботи. Високий рівень імітації припускає повну реалізацію всіх функціональних можливостей сервісу (по суті, відбувається наближення до емуляції сервісу). При використанні реальних операційних систем рівень імітації – високий. У дану характеристику також включається рівень імітації поведінки при реакції на спробу атаки.

Рівень ризику – характеризує ступінь ризику при використанні Honeypot. Чим більше функціональних можливостей надає Honeypot, тем вище ймовірність, що Honeypot може бути використаний для атаки інших систем або сервісів. Низький рівень ризику містить у собі можливу атаку тільки проти імітуємого сервісу. Середнім рівнем ризику мають засоби Honeypot, що імітують кілька сервісів одночасно. Високий рівень ризику – недолік Honeypot сильної взаємодії.

У цей час існує класифікація засобів Honeypot по ступеню взаємодії зі зловмисником [1,4]. Виділяються три основні види Honeypot:

- слабкої взаємодії;
- середньої взаємодії;
- сильної взаємодії.

Кожний із цих видів Honeypot надає певну функціональність, або рівень взаємодії зловмисника із системою. Таким чином, починаючи від Honeypot слабкої взаємодії, функціональні можливості Honeypot розширюються, що й показують перехідні стрілки. Разом з функціональними можливостями змінюються й значення раніше наведених ознак першої класифікації для кожного виду Honeypot.

Honeypot слабкої взаємодії

Засоби Honeypot слабкої взаємодії звичайно достатньо просто встановити, настроїти, використовувати й підтримувати, тому що вони мають просту структуру й базову функціональність. Як правило, дані Honeypot імітують тільки частину сервісів. Зловмисник обмежений у взаємодії із цими сервісами. Наприклад, Honeypot слабкої взаємодії може імітувати стандартний сервер Unix з декількома запущеними сервісами, такими як telnet або FTP. Зловмисник може

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		23

створити з'єднання telnet до системи, одержати баннер операційної системи й запит login. Після чого, зловмисник може робити спроби добору пароля, які будуть записуватися Honeypot, але справжньої операційної системи ні, і взаємодія зловмисника із системою на даному етапі закінчується.

Іншим прикладом може служити імітуємий Ftp-сервер, на який зловмисник може ввійти під анонімним обліковим записом і скачати файл із пароллями. При цьому існує тільки один обліковий запис – анонімна. І файл, природно, буде не сьогоденням. Взаємодія з Honeypot обмежене спробами зробити авторизацію, анонімним доступом і здатністю скачати підроблений файл із пароллями.

Головна мета Honeypot слабкої взаємодії – виявлення, особливо, сканувань і несанкціонованих спроб з'єднання. У силу того, що дані засоби Honeypot надають обмежену функціональність, більшість із них представляються у вигляді програм. Програма може бути просто встановлена на хост і налаштована відповідно до вимог. Завдання адміністратора в цьому випадку – робити моніторинг попереджень, які генерує Honeypot, а також відслідковувати зміни імітуемого програмного забезпечення.

Тому що Honeypot слабкої взаємодії прості, вони мають мінімальний рівень протоколювання й найменший рівень ризику. Низька функціональність приводить до зменшення можливих дій для протоколювання й низької ймовірності припуститися помилки. Крім того, реальна операційна система не надається для взаємодії зловмисникові, тому Honeypot даного типу не може бути використаний для атаки або дослідження інших систем. Ризик також зменшує той факт, що Honeypot слабкої взаємодії досить легко використовувати й підтримувати.

Засоби Honeypot подібного виду обмежені в кількості інформації, яку вони можуть надати про зловмисника. Застосунок слабкої взаємодії надають транзакційну інформацію. Транзакційна інформація – це дані, зібрані про

обставини атаки, але не про саму атаку. Для Honeypot слабкої взаємодії це наступне:

- час і дата атаки;
- IP-адреса й порт джерела (зловмисника);
- IP-адреса й порт призначення (засоби Honeypot).

Якщо імітуємі сервіси надають яка-небудь взаємодія зі зловмисником, то існує можливість одержати його активність. Який обсяг активності може бути отриманий, залежить від можливостей і рівня імітації Honeypot слабкої взаємодії.

Honeypot слабкої взаємодії спрямовані на збір відомої поведінки. Зловмисник взаємодіє звичайним образом, а Honeypot має заздалегідь певні види відповідей. Honeypot слабкої взаємодії показують поганий або навіть негативний результат при взаємодії з невідомими або непередбаченими видами атак.

У силу їх простоти й невеликого ризику, Honeypot слабкої взаємодії рекомендується для індивідуального використання або для невеликих організацій. Honeypot слабкої взаємодії також можуть бути використані для поліпшення розуміючи даної технології.

Honeypot середньої взаємодії

Засіб Honeypot середньої взаємодії надає зловмисникові більше можливостей, ніж Honeypot слабкої взаємодії, але має меншу функціональність у порівнянні з Honeypot сильною взаємодії. Дані засоби можуть очікувати різну активність, і спроектовано давати кілька можливих відповідей на дії зловмисника.

Наприклад, якщо імітується IIS Web Server, те Honeypot може бути зконфігурований з різною функціональністю й поведінкою, щоб виявляти певного мережного хробака. Даний рівень взаємодії вже вище, ніж в Honeypot слабкої взаємодії, де є присутнім тільки Http-баннер. При цьому є можливість зібрати поведінку хробака для наступного аналізу. Йому не надається повноцінна операційна система, що знижує ризик. Існує тільки імітуємий застосунок.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		25

Іншим прикладом Honeypot середньої взаємодії є деякі можливості операційної системи Unix, такі як “chroot”, або ж в Windows – Virtual Machine Ware (VMWare). Вони дозволяють адміністраторові розбивати оточення операційної системи, створюючи віртуальні операційні системи усередині реальної. Віртуальна операційна система може контролюватися реальною операційною системою, але використання такої системи зовні дуже схоже зі справжньої. Зміст полягає в тому, що зловмисник буде взаємодіяти з підробленим оточенням, таким чином, уся його активність може бути контрольована з боку реальної операційної системи.

Однак даний підхід має кілька проблем. По-перше, даний застосунок достатній складне, таким чином, на стадії роботи або конфігурування можуть виявитися помилки. По-друге, додати віртуальному оточенню повну функціональність і взаємодія з реальною операційною системою – трудомістке завдання. Чим більше функціональності й реалістичності надається віртуальному оточенню, тем легше для зловмисника обійти дане оточення й одержати контроль над реальною операційною системою. Таким чином, більшість Honeypot середньої взаємодії не надають повній функціональності стандартної операційної системи.

Honeypot середньої взаємодії вимагає кілька більших зусиль при установці й налаштуванні, чому слабкої взаємодії. Звичайно дані застосунки не поставляються у вигляді готових програмних застосунків. Для того щоб увести в експлуатацію подібний вид Honeypot потрібна детальне налаштування функціональності в порівнянні з Honeypot слабкої взаємодії.

Використання й підтримка засобів Honeypot середньої взаємодії також більш складний процес. Зловмисники мають більшу частку взаємодії із системою, тому потрібно здійснити додаткові заходи безпеки. Необхідно розробити механізми, які дадуть упевненість, що зловмисник не зможе ушкодити іншим системам, і збільшена функціональність не виявиться вразливою. Крім того,

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		26

повинна здійснюватися регулярна підтримка Honeypot, що виражається в постійному моніторингу нових уразливостей.

Honeypot середньої взаємодії також більш складні й збільшують ступінь ризику. При цьому вони здатні одержати набагато більше інформації. На відміну від звичайного сканування портів, існує можливість запису активності мережних хробаків і зловмисників, вивчення стану системи після злому (наприклад, яким чином зловмисник проникнув у систему й підвищив права), а також у зборі програмних засобів зловмисника.

Honeypot сильної взаємодії

Засоби Honeypot сильної взаємодії досить небезпечні. Вони надають більшу кількість інформації про зловмисників, але вимагають достатній час для побудови й підтримки. Крім того, приносять найвищий рівень ризику. Ціль Honeypot сильної взаємодії – надати зловмисникові доступ до реальної операційної системи, де нічого не імітується або обмежується. Зручності для вивчення в цьому випадку неймовірні. Існує можливість досліджувати нові засоби, виявляти нові уразливості в операційних системах або застосунках, а також пізнавати, яким чином зловмисники зв'язуються між собою.

Для створення подібного оточення, у принципі, не потрібно вносити які-небудь зміни в реальні операційні системи. Більшість стандартних складань не мають ніяких відмінностей щодо існуючих виробничих систем у більшості організацій. Єдина річ, яка визначає ці системи як Honeypot, – це те, що вони не мають виробничого значення. Їх ціль – бути дослідженими, атакованими або скомпрометованими. Слід зробити вивід, що такий потужний засіб приносить величезний рівень ризику. Після одержання зловмисником повного доступу, він починає взаємодіяти з повнофункціональною операційною системою, яка надає йому можливість здійснювати будь-які дії, наприклад, атакувати інші системи або збирати внутрішній трафік. Великий обсяг роботи повинен полягати в зниженні цих ризиків.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		27

У більшості випадків Honeypot сильної взаємодії розташовуються в контрольованому середовищі, наприклад, у мережі – після міжмережевого екрана. Здатність контролювати зловмисника відбувається не від самого Honeypot, а від контролюючого мережного пристрою – найчастіше, міжмережевого екрана. Міжмережевий екран надає зловмисникові можливість атакувати засіб Honeypot, але забороняє робити зовнішні атаки. Тому що побудована архітектура досить складна, то потрібно детально визначити базу правил міжмережевого екрана.

Через розширений механізм контролю, установка й налаштування Honeypot сильної взаємодії вимагає достатніх зусиль і часу. Крім того, різні технології забезпечення безпеки, наприклад, раніше розглянуті міжмережеві екрани, а також системи виявлення вторгнень, можуть бути також включені в побудову Honeypot сильної взаємодії. Однак підтримка даної системи буде ускладнена відновленням баз правил міжмережевого екрана й сигнатур атак системи виявлення вторгнень. При цьому, чому більша взаємодія дозволяється зловмисникові, тим більше ймовірність помилки на якомсь кроці. Але один раз коректно встановлений і налаштований, Honeypot сильної взаємодії може надати таку інформацію, яку не зможе дати жоден з інших типів Honeypot.

Кожний з розглянутих видів засобів Honeypot має відповідні значення ознак по першій класифікації.

Функціональні вимоги до Honeypot

С обліком побудованої класифікації висуваються наступні функціональні вимоги до засобів Honeypot:

- Можливість вибору варіанта імітуємого сервісу (наприклад, вибір різного Ftp-сервера).
- Максимальне наближення до імітуємому сервісу:
 - особливості роботи;
 - реакція на стандартні/нестандартні команди або запити;
 - підтримка реалізованих додаткових можливостей обраного сервісу.

- Існування можливості (або умов) емуляції злому сервісу.
- Відповідність поточному (сучасному) набору уразливостей. Під набором уразливостей розуміється сукупність відомих уразливостей для даного сервісу, а також набір варіантів аномальної поведінки при взаємодії з імітуємим сервісом. Таким чином, Honeyrot повинен містити набори уразливостей для кожного імітуємого сервісу й аналітичний модуль прийняття застосунків для встановлення факту злому або наявності аномальної поведінки при взаємодії з імітуємого сервісом.
- Існування бази даних для зберігання наборів уразливостей, з можливістю зміни й доповнення.
- Захист БД наборів уразливостей від додаткових погроз (можливість реалізації атаки на інші, не імітуємі сервіси, у результаті якої властивості цілісності й конфіденційності БД наборів уразливостей будуть порушені). Одним з головних вимог по даному пункту є вимога розміщення БД уразливостей на іншому хості.
- Повне протоколювання взаємодії потенційного порушника з Honeyrot. Протокол повинен містити як інформацію, яка була отримана від порушника, так і відповідь імітуємого сервісу, який був відісланий порушникові.

3.2 Розробка структурної схеми

Honeynet: мережа Honeyrot

Honeyrot – це мережа-приманка, яка містить один або декілька honeypot-ів. Вона виглядає як справжня мережа та містить кілька систем, але розміщена на одному або лише кількох серверах, кожен з яких представляє одне середовище. Наприклад, honeypot-машина Windows, honeypot-машина Mac та honeypot-машина Linux. «Медова стіна» відстежує трафік, що входить і виходить з мережі, і спрямовує його до екземплярів honeypot. Ви можете впроваджувати вразливості в honeynet, щоб полегшити зловмиснику доступ до пастки.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		29

Приклад топології Honeynet

Будь-яка система в мережі Honeynet може служити точкою входу для зловмисників. Honeynet збирає розвідувальні дані про зловмисників і відводить їх від реальної мережі. Перевага honeynet над простою мережею Honeuprot полягає в тому, що вона більше схожа на реальну мережу та має більшу зону охоплення.

Це робить honeynet кращим рішенням для великих, складних мереж – він надає зловмисникам альтернативну корпоративну мережу, яка може бути привабливою альтернативою реальній.

Пастка для спаму: приманка електронної пошти

Спам-пастки – це інструменти боротьби з шахрайством, які допомагають інтернет-провайдерам (ISP) виявляти та блокувати спамерів. Вони допомагають зробити вашу поштову скриньку безпечнішою, блокуючи вразливості. Спам-пастка – це фальшива електронна адреса, яка використовується для приманки спамерів. Малоімовірно, що легітимна пошта буде надіслана на фальшиву адресу, тому, коли електронний лист отримано, це, найімовірніше, спам.

Типи спам-пасток включають:

- Орфографічні помилки в імені користувача – спам-фільтр виявляє орфографічні помилки, що виникли внаслідок людської або машинної помилки, зокрема, та надсилає електронний лист до папки «Спам». Це стосується й адрес електронної пошти з помилками, наприклад, jhon@labra.com замість справжньої john@labrat.com.

- Термін дії облікових записів електронної пошти закінчився – деякі провайдери використовують покинуті облікові записи електронної пошти або доменні імена, термін дії яких закінчився, як пастки для спаму.

- Куплені списки розсилки електронною поштою – вони часто містять багато недійсних адрес електронної пошти, що може стати причиною спам-пастки. Крім того, оскільки відправник не отримав дозволу на надсилання електронних листів обліковим записам у списку, його можуть вважати спамерами та внести до чорного списку.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		30

Вразливості спам-пасток включають генерування зворотного розсіювання (неправильно автоматизованих повідомлень про відмову) та інфікування легітимних адрес електронної пошти, які відповідають на повідомлення або пересилають його.

Більше того, як тільки спам-пастка викрита, спамери можуть її використати, надсилаючи на неї легітимний контент, що призводить до втрати ефективності спам-пастки. Ще один ризик полягає в тому, що деякі люди можуть писати на адресу, не усвідомлюючи, що це спам-пастка.

Випадкове потрапляння в спам-пастку може зашкодити вашій організації, вплинувши на вашу репутацію та якість доставки. Інтернет-провайдер може заблокувати або внести до чорного списку вашу IP-адресу, а компанії, які звертаються до антиспам-баз даних, фільтруватимуть ваші електронні листи.

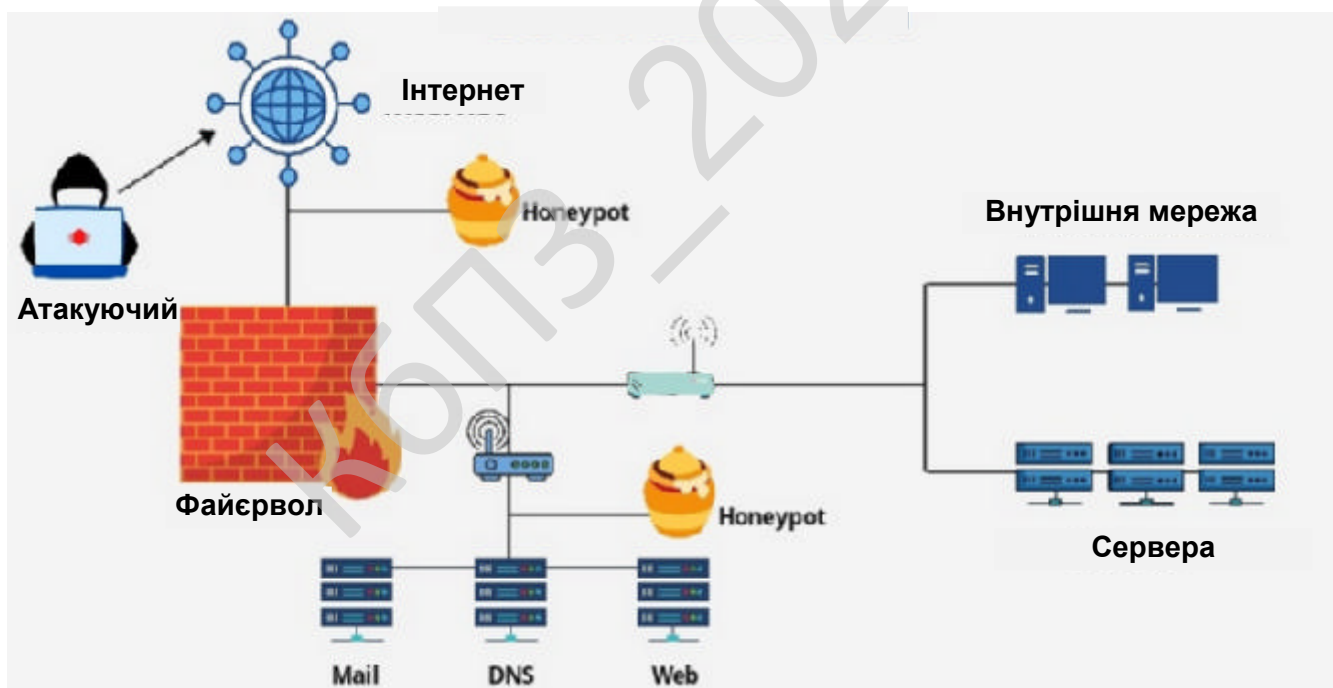


Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

На функціональній схемі, що наведена на рисунку 3.2, показана реалізація запропонованого методу. Наведено результати виконання окремих етапів. Інтерес представляє ієрархічна організація баз шаблонів системи кібербезпеки для автоматизованого управління системами типу Honeypot. Порівняння відбувається за принципом поступового наближення зі знаходження статистичної залежності чорно-білих зображень на основі кореляційного методу.

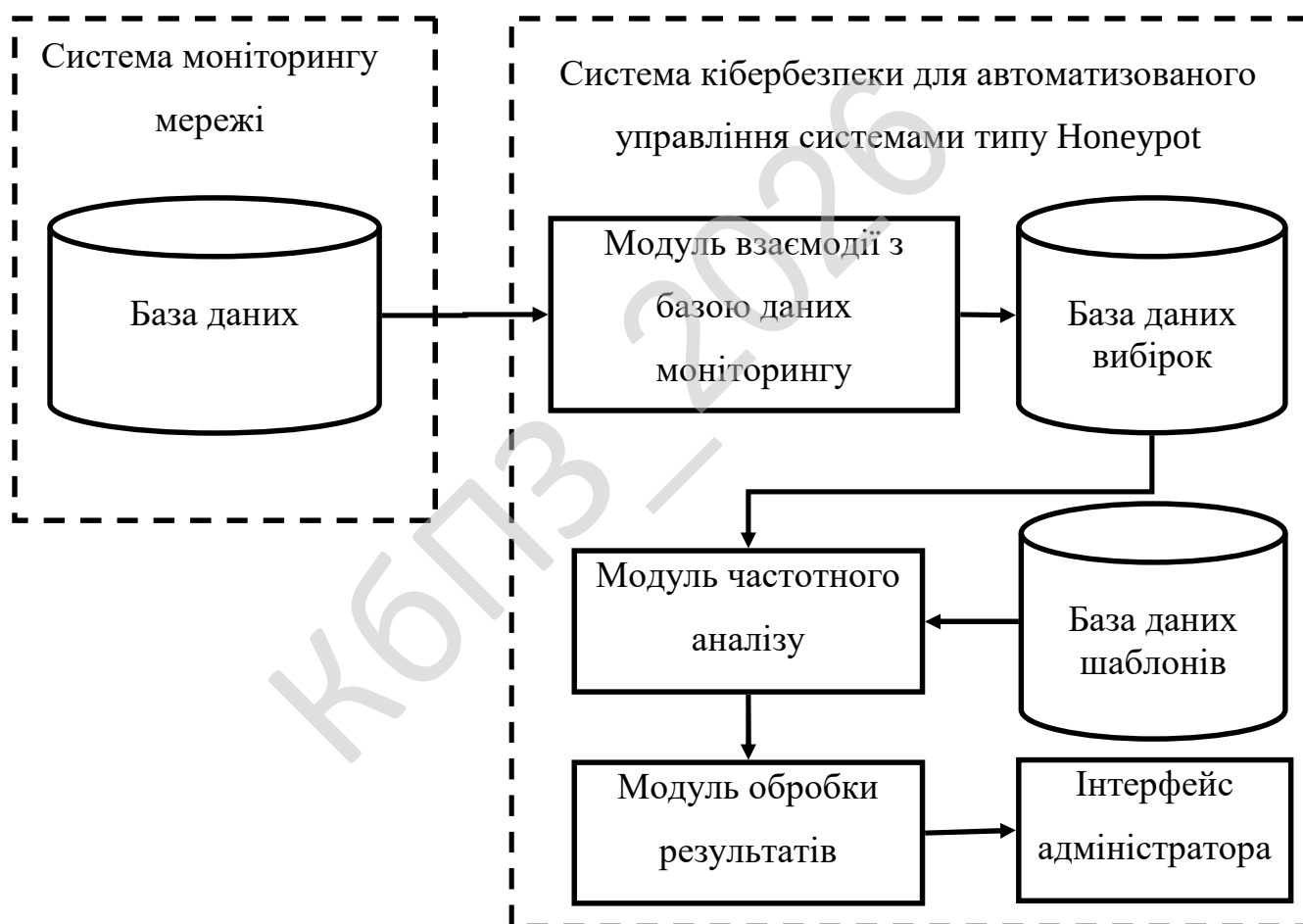


Рисунок 3.2 – Функціональна схема системи

При цьому розмірність еталона й досліджуваного подання по вертикалі збігаються. Рух еталона уздовж вейвлет-образу сигналу відбувається в

горизонтальному напрямку у випадку стаціонарного аналізу. При динамічному виявленні особливостей еталон і сигнал зіставляються правими границями.

Поява аномалій у різних точках системи кібербезпеки для автоматизованого управління системами типу Honeypot спостереження протягом невеликого проміжку часу говорить про можливий взаємозв'язок подій. Об'єкт, що має більше ранній за часом момент виникнення аномалії може бути джерелом нестандартного поведіння ряду даних. Досліджуючи виникнення аномалій на різних пристроях, можливим є побудова дерева аномалій, що веде від джерела через посередників різних рівнів до приймачів і навпаки: від приймача до джерела.

Для реалізації функціональної схеми опишемо структуру методу динамічного виявлення аномалій в інтенсивностях потоків телекомунікаційних даних системи кібербезпеки для автоматизованого управління системами типу Honeypot, що включає наступні основні етапи:

1. Збір даних.
2. «Ручна» класифікація частини даних на предмет присутності нормального (типового) або аномального трафіку.
3. Попередній Фур'є-аналіз типового трафіку.
4. Виконання вейвлет-розкладання типового трафіку за обраними функціями для оцінки його спектрального-часового подання.
5. Обробка вейвлет-подання аномального трафіку.
 - а) визначення моментів різкої зміни характеру поведінки рядів телекомунікаційних даних по рівневим і частотним характеристиках;
 - б) формування баз шаблонів вейвлет-образів:
 - бази вейвлет-образів для характеру розвитку трафіку на тлі критичних ситуацій;
 - бази вейвлет-образів для характеру розвитку трафіку до критичних моментів (до початку зареєстрованих проблем у роботі мережі);
 - збереження відповідностей між даними баз;

в) визначення частотних показників для провокуючих розвиток аномалії подій системи кібербезпеки для автоматизованого управління системами типу Honeypot, у тому числі аналіз енергетичних спектрів послідовностей до критичних моментів (спектра потужності й скалограми) на предмет перерозподілу енергії між частотами;

6. Верифікація прогнозу характеристик трафіку з реальними значеннями (на основі попередньо зібраних даних поза реальним режимом часу).

7. При надходженні нових даних:

а) зіставлення їх вейвлет-подань із шаблонами сформованих баз даних за допомогою кореляції й формування рішення про навісність передкритичної або критичної ситуації.

б) виявлення зв'язків між аномаліями, що відбулися в різних точках спостереження.

У результаті виконання етапів з першого по шостий включно маємо:

– алгоритм розпізнавання позаштатної ситуації системи кібербезпеки для автоматизованого управління системами типу Honeypot;

– початкові бази вейвлет-шаблонів для позаштатних ситуацій і попередніх за часом інтервалів;

– модель прогнозування характеру розвитку трафіку і як наслідок апріорне визначення можливих позаштатних ситуацій і їхня класифікація.

Застосування спектральних і спектрально-часових алгоритмів, є основними інструментами в розв'язуваному завданні динамічного виявлення аномалій у часових рядах інтенсивностей телекомунікаційних даних:

– аналіз внесків частот визначає формування моделей типового трафіку й окремих ділянок з особливостями;

– картини вейвлет-коефіцієнтів аномалій і попередніх їм ділянок сигналу формують бази даних шаблонів для характеру розвитку трафіку на тлі критичних ситуацій і до критичних моментів;

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		34

– побудовані на основі обчислених коефіцієнтів енергетичні спектри й інші залежності дозволяють відслідковувати розподіл (або перерозподіл) енергії між частотами й визначати сховану природу окремих ділянок сигналу.

Розкрито етапи нагромадження інформації, аналізу готових наборів, формування баз даних шаблонів, верифікації, виявлення аномалій у потоках даних, аналізу взаємозв'язків аномалій у різних потоках.

Детально проаналізовані алгоритми дискретного перетворення Фур'є (ДПФ), дискретного перетворення Хартлі (ДПХ), швидкого перетворення Фур'є (ШПФ) і швидкого перетворення Хартлі (ШПХ).

Обґрунтовано можливість заміни широко використовуваного ДПФ на менш відоме, але більше ефективне ДПХ.

У результаті отримані математичні моделі оптимізації короточасного аналізу (за участю часового «вікна» сканування) для прямих ПФ і ПХ зсувної послідовності.

Отримані практичні результати підтвердили ефективність запропонованих методів перетворення й у своєму загальному виді збіглися з теоретичними, незважаючи на присутні відхилення, внесені специфікою програмної реалізації. Відповідно до отриманих значень середніх часів виконання короточасного спектрального аналізу в межах одного «вікна» можна зробити наступний висновок: для скорочення часу виконання короточасного Фур'є-аналізу треба перше «вікно» обчислювати за допомогою ШПХ із наступним перетворенням обчислених значень у результат ПФ, а наступні «вікна» на основі запропонованого методу для ПФ.

Переваги запропонованого методу:

- висока швидкість обчислень;
- лінійна залежність швидкості обчислень від розміру «вікна».

Недоліки запропонованого методу:

- наявність швидко зростаючої погрішності обчислень ПХ після другої ітерації алгоритму перерахунку спектра.

Для усунення недоліку запропонованого методу необхідно:

- застосування розширених форматів подання даних;
- при досягненні критичного значення погрішності зробити перерахування спектра по стандартному алгоритмі.

Далі розглянемо спектрально-часові вейвлет-перетворенням (ВП). Розглянемо принципи різних варіацій дискретизованого перетворення, вплив базису на результат вейвлет-розкладання й дані рекомендації з можливого використання вейвлетів.

Докладно розглянуті:

- 1) дискретне перетворення зі зміною масштабів і зрушень за значенням, рівному часової локалізації вейвлета;
- 2) дискретний діадний кратномасштабний аналіз;
- 3) дискретизоване розкладання.

Виконано аналіз виникаючих крайових ефектів спектрально-часових картин ВП на прикладах розгляду кутів впливу елементів аналізованої послідовності на вейвлет-коефіцієнти й кутів вірогідності (правдоподібності) у сітках перетворень. У середині кута правдоподібності, як відомо, крайові ефекти проявлятися не можуть, оскільки кожна особливість, у тому числі й виникаюча на краях аналізованого сигналу, робить лише локальний вплив на вейвлет-спектрограму. Скоректовано від кутів впливу й правдоподібності для зсувної послідовності.

З позиції інформативності по кількості уточнюючих коефіцієнтів кращим є випадок 3), потім 2), потім 1).

З погляду збільшення області достовірних значень кращим є випадок 1), потім 3), потім 2).

У порядку зростання обчислювальної складності перетворення розташовуються в порядку 1), 2), 3).

Для завдання розпізнавання аномалії перші два критерії є вирішальними, причому по жодному з них обране перетворення не повинне бути гіршим.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		36

Таким є дискретизоване перетворення, хоча його надмірність очевидна й підтверджується третім критерієм – обчислювальною складністю.

Можна припустити, що оптимальним було б перетворення, «яке знаходиться» між дискретним діадним і дискретизованим, причому саме перше повинне бути взяте за основу, так як має добре оптимізовані по організації обчислень і використовуваної пам'яті швидкі алгоритми статичного обчислення сітки коефіцієнтів, і розширено для наділення достоїнствами дискретизованого розкладання. Результат може бути аналогічний використанню фреймів, так як перетворення на основі фреймів має велика подібність із дискретизованим розкладанням з тією різницею, що зміна масштабу відбувається по ступенях «двійки».

Розроблено схему виконання дискретизованого за часом і діадного по масштабах розкладання на основі швидкого алгоритму діадного кратномасштабного аналізу. Відзначимо, що в цей час існують алгоритми, оптимізуючі стандартну схему швидкого діадного розкладання, наприклад на основі нестандартного матричного множення, тобто пропонується у роботі метод розрахунку вейвлет-коефіцієнтів може бути додатково оптимізований по обчислювальних витратах.

Виконано оптимізацію обчислень сіток дискретизованого розкладання за принципом фреймів і покладених у його основу алгоритмів діадного кратномасштабного аналізу при зрушенні послідовності.

Розроблено підхід до формування довірчої області на картині вейвлет-коефіцієнтів. Завдання було вирішено послідовним виконанням наступних етапів:

- вибір алгоритму пророкування значень майбутніх елементів часового ряду;
- розрахунок довірчого інтервалу для передвіщених значень;
- виконання верифікації;
- установлення зв'язку довірчого інтервалу із границею кута правдоподібності;

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		37

– розширення кута правдоподібності за допомогою введення довірчої області в картину вейвлет-коефіцієнтів.

Запропоновано підхід для мінімізації розкиду значень вейвлет-коефіцієнтів у розрахованій довірчій вейвлет-області на основі введення нестандартної внутрішньої границі довірчої часової області.

На впевненість влучення вейвлет-коефіцієнтів у розрахований діапазон (вейвлет-упевненість) впливають значення відрахунків вейвлета, так як їх можна інтерпретувати в рамках справжнього етапу як вагові коефіцієнти, відповідно до яких приймаються в розгляд відрахунки часового ряду, у тому числі неіснуючі.

Для використання в аналізі довірчої вейвлет-області найбільш підходящим є формування декількох її подань, заповнених, наприклад, значеннями:

- верхньої границі;
- нижньої границі;
- розрахованими на основі передвіщених значень апроксимованої часової послідовності;
- випадково розподіленими між верхньою й нижньою границями.

Комбінація таких подань довірчої вейвлет-області дозволить найбільше вірно судити про поводження вейвлет-коефіцієнтів за межами кута правдоподібності.

Для одержання картини вейвлет-коефіцієнтів, придатних для виконання операцій розпізнавання, необхідна певна довжина вихідної послідовності. Мала довжина послідовності спричиняється пропорційну кількість вейвлет-коефіцієнтів, що робить картину розкладання розбитої на відповідну кількість прямокутних областей. Елементи такої картини занадто грубі й непридатні для подальшого розпізнавання.

До вихідної послідовності пропонується застосувати алгоритм, що збільшує число елементів ряду. Проаналізовано результати використання спектральних і часових алгоритмів інтерполяції.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		38

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

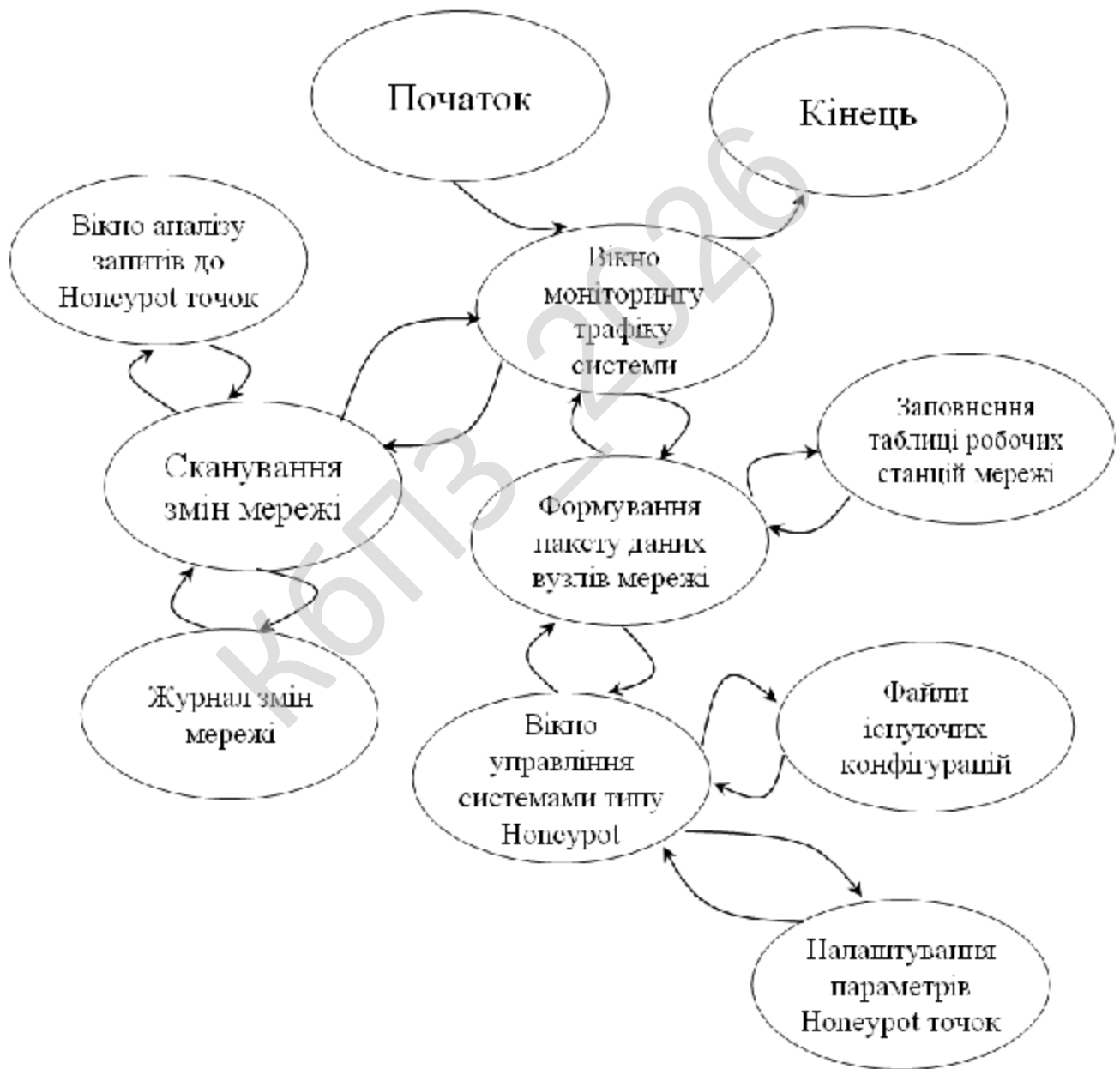


Рисунок 3.3 – Діаграма взаємодії процесів

Основні складові елементи діаграми взаємодії процесів це потоки даних:

- Репозиторії, потік сховища даних.
- Потоки зовнішні по відношенню до системи сутності.
- Процеси які являють собою трансформацію даних в рамках описуваної системи.
- Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1.

З рисунку видно, що після запуску програми спочатку відбувається вивід основного вікна програми. Потім здійснюється наступна послідовність дій:

– Виведення вікон автоматизованого управління системами типу Honeypot.

– Формування точки Honeypot з даних по замовчанню.

– Внесення змін користувачем.

– Підпрограма створення точки Honeypot.

– Формування шаблонів мережі.

– Збереження шаблонів у базі даних ПЗ.

– Збір даних поточного стану.

– Відображення інформації про мережу.

– Відображення часових рядів трафіку.

– Підпрограма розпізнавання вторгнень.

– Виведення інформації про виявлення аномального трафіку.

– Блокування аномального трафіку.

Робота підпрограми:

– Аналіз трафіку та порівняння його з шаблонами точок Honeypot.

– Автоматично відмітити трафік як аномальний.

– Розпізнавання типу позаштатної ситуації.

– Формування рішення про наявність передкритичної або критичної

ситуації.

- Виявлення зв'язків між аномаліями, що відбулися в різних точках спостереження.
- Запис до журналу подій про позаштатну ситуацію.
- Прогнозування характеру розвитку поведінки трафіку.
- Формування та виведення звіту.

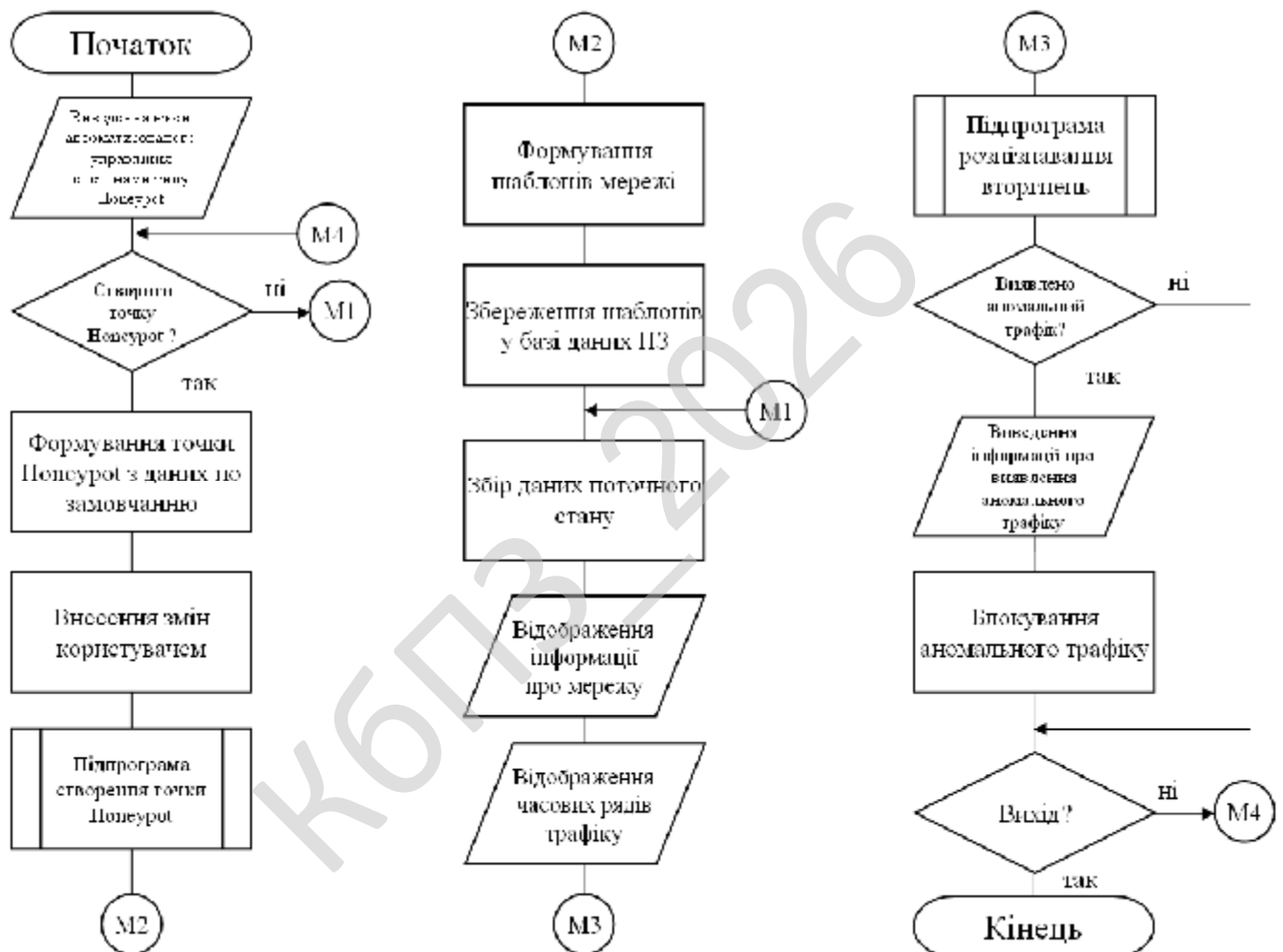


Рисунок 4.1 – Блок-схема основної програми

Опис архітектури системи

Програмне забезпечення системи автоматизованого управління системами типу Honeypot реалізується мовою Python і призначається для виявлення

підозрілої мережевої активності шляхом імітації сервісів, які можуть зацікавити порушника. Система не надає реального доступу до ресурсів, а приймає підключення, фіксує параметри взаємодії, аналізує активність і формує журнал подій.

Система має модуль керування приманками, модуль мережевого прослуховування, модуль журналювання, модуль аналізу активності, модуль бази даних і модуль звітності. Користувач задає список портів і назв імітованих сервісів. Програма запускає окремі обробники для кожного сервісу, приймає підключення, зберігає IP адресу, порт, час події, отримані дані та рівень ризику.

Основні функції системи:

- Система виконує запуск і зупинку Honeypot сервісів;
- Система приймає підключення до контрольованих портів;
- Система записує події у базу SQLite;
- Система визначає рівень ризику за частотою звернень і змістом отриманих даних;
- Система формує короткий звіт для адміністратора.
- Система дозволяє централізовано керувати кількома віртуальними приманками.

Наведемо частину коду, яка реалізує деякі з вищеперерахованих функцій.

```
import asyncio
import sqlite3
import datetime
from dataclasses import dataclass

#Модель налаштування окремої Honeypot приманки
@dataclass
class HoneypotService:
    name: str
    host: str
    port: int
    banner: str

#Модуль збереження подій у базі даних
class EventStorage:
    def __init__(self, db_name="honeypot_events.db"):
```

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		43

```

        self.db_name = db_name
        self.create_table()
    def create_table(self):
        with sqlite3.connect(self.db_name) as connection:
            cursor = connection.cursor()
            cursor.execute("""
                CREATE TABLE IF NOT EXISTS events (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    service_name TEXT NOT NULL,
                    source_ip TEXT NOT NULL,
                    source_port INTEGER NOT NULL,
                    target_port INTEGER NOT NULL,
                    payload TEXT,
                    risk_level TEXT NOT NULL,
                    created_at TEXT NOT NULL
                )
            """)
            connection.commit()
    def save_event(self, service_name, source_ip, source_port, target_port,
payload, risk_level):
        created_at = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        with sqlite3.connect(self.db_name) as connection:
            cursor = connection.cursor()
            cursor.execute("""
                INSERT INTO events (
                    service_name,
                    source_ip,
                    source_port,
                    target_port,
                    payload,
                    risk_level,
                    created_at
                )
                VALUES (?, ?, ?, ?, ?, ?, ?)
            """, (
                service_name,
                source_ip,
                source_port,
                target_port,
                payload,
                risk_level,
                created_at
            ))

```

```

        ))
        connection.commit()
    def get_report(self):
        with sqlite3.connect(self.db_name) as connection:
            cursor = connection.cursor()
            cursor.execute("""
SELECT service_name, source_ip, target_port, risk_level, created_at
FROM events
ORDER BY id DESC
LIMIT 20
""")
            return cursor.fetchall()

#Модуль аналізу підозрілої активності
class RiskAnalyzer:
    def analyze(self, payload):
        suspicious_words = ["admin", "root", "password", "login", "cmd",
"select", "drop"]
        payload_lower = payload.lower()
        for word in suspicious_words:
            if word in payload_lower:
                return "високий"
        if len(payload) > 100:
            return "середній"
        return "низький"

#Модуль керування Honeypot сервісами
class HoneypotManager:
    def __init__(self, services):
        self.services = services
        self.storage = EventStorage()
        self.analyzer = RiskAnalyzer()
    async def handle_client(self, reader, writer, service):
        address = writer.get_extra_info("peername")
        source_ip = address[0]
        source_port = address[1]
        writer.write(service.banner.encode("utf-8"))
        await writer.drain()
        try:
            data = await asyncio.wait_for(reader.read(1024), timeout=10)
            payload = data.decode("utf-8", errors="ignore")
        except asyncio.TimeoutError:

```

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

```

        payload = "Підключення без передавання даних"
        risk_level = self.analyzer.analyze(payload)
        self.storage.save_event(
            service.name,
            source_ip,
            source_port,
            service.port,
            payload,
            risk_level
        )
        response = "Access denied\r\n"
        writer.write(response.encode("utf-8"))
        await writer.drain()
        writer.close()
        await writer.wait_closed()
    async def start_service(self, service):
        server = await asyncio.start_server(
            lambda reader, writer: self.handle_client(reader, writer, service),
            service.host,
            service.port
        )
        print(f"Запущено сервіс {service.name} на порту {service.port}")
        async with server:
            await server.serve_forever()
    async def start_all(self):
        tasks = []
        for service in self.services:
            task = asyncio.create_task(self.start_service(service))
            tasks.append(task)
        await asyncio.gather(*tasks)
    def print_report(self):
        rows = self.storage.get_report()
        print("Останні події Honeypot")
        for row in rows:
            print(row)

def main():
    services = [
        HoneypotService(
            name="SSH приманка",
            host="127.0.0.1",
            port=2222,

```

```

        banner="SSH-2.0-OpenSSH_8.4\r\n"
    ),
    HoneypotService(
        name="HTTP приманка",
        host="127.0.0.1",
        port=8080,
        banner="HTTP/1.1 200 OK\r\nServer: Apache\r\n\r\nFake admin
panel\r\n"
    )
]
manager = HoneypotManager(services)
try:
    asyncio.run(manager.start_all())
except KeyboardInterrupt:
    print("Роботу системи завершено")
    manager.print_report()
if __name__ == "__main__":
    main()

```

Незважаючи на те що я працював над ПЗ один в реалізації програми я використовував підходи пришвидшення розробки на основі методологій Agile – Extreme Programming.

Екстремальне програмування (Extreme Programming, далі XP) це методологія розробки програмного забезпечення, найпопулярніша серед так званих гнучких методологій. Має на меті поліпшення якості програмного забезпечення та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються. Авторами даної методології є Кент Бек, Ворд Каннінгем, Мартін Фаулер та інші.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		47

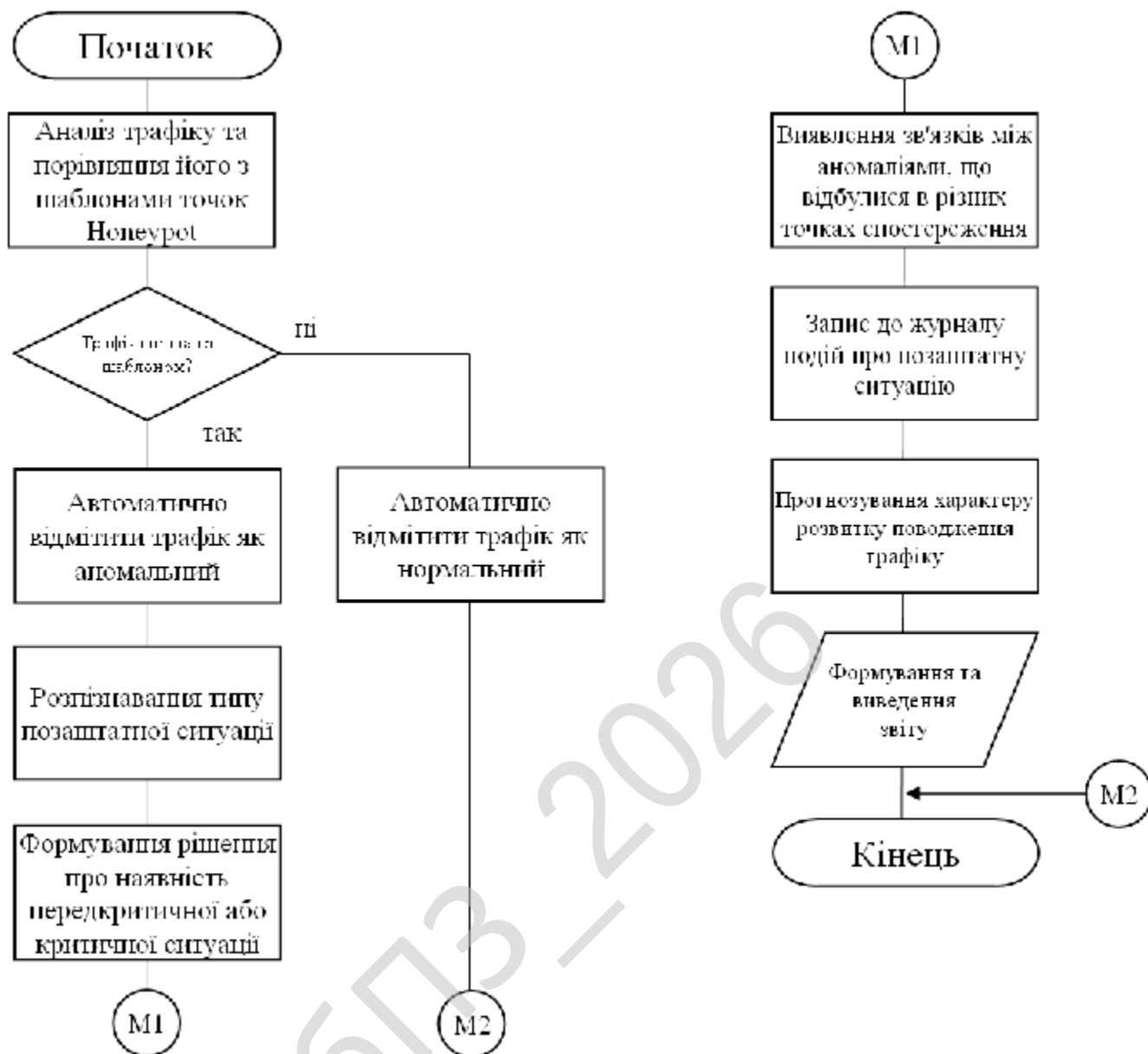


Рисунок 4.2 – Блок-схема роботи підпрограми

Інші елементи екстремального програмування включають в собі: парне програмування, проведення обширної перевірки сирцевого коду, модульне тестування всього коду, уникання створення функціональності до того як вона дійсно необхідна, простота та ясність коду, очікування на зміну вимог замовників з плином часу та коли вимоги до продукту стають ясніші, досить часте спілкування із замовником та між самими програмістами.

Назва методології походить від ідеї застосувати корисні методи і практики розробки програмного забезпечення, піднявши їх до "екстремальних" рівнів.

Критики ХР зауважують на потенційні недоліки цієї методології – нестабільні вимоги, незадокументовані компроміси конфліктів користувачів, відсутність загального документу дизайну програми.

Хоча саме екстремальне програмування є відносно новим, багато її практик вже існували і використовувались протягом певного часу; однак, методологія підносить "найкращі практики" до екстремального рівня. Для прикладу, практика по плануванню і написанню тестів перед написанням кожної маленької частини коду було використано раніше в проекті НАСА "Меркурій". Для зменшення часу на розробку ПЗ деякі формальні документи тестування (такі як приймальне тестування) писались паралельно (або й раніше) з написанням самого ПЗ. Незалежна група тестування НАСА може писати процедури тестування базуючись на формальних вимогах до продукту до того як програмне забезпечення розроблене та інтегроване в систему. В ХР ця концепція піднесена до "екстремального рівня" завдяки написанню автоматичних тестів які перевіряють поведінку навіть малих частинок коду, а не тільки значних функціональних частин ПЗ.

Посібник Extreme Programming Explained: Embrace Change описує Екстремальне Програмування, як:

- Спроба примирити гуманність і продуктивність.
- Механізм для соціальної зміни.
- Шлях до удосконалення.
- Стиль розвитку.

Дисципліна розробки програмного забезпечення.

Головною метою Екстремального Програмування є скорочення вартості неочікуваних змін. У традиційних методах розробки (на кшталт SSADM) вимоги до розвитку системи визначаються на початку роботи над проектом, і часто виправляються пізніше. Це означає, що вартість проекту через зміни буде більшою

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		49

за заплановану (традиційна особливість для програмного забезпечення, що проектується).

ХР використовується для скорочення вартості змін, завдяки представленню простих значень, принципів і методів. При використанні екстремального програмування, проект повинен стати гнучкішим щодо змін.

Extreme Programming Explained описує екстремальне програмування як дисципліну розробки програмного забезпечення яка змушує людей створювати високоякісне ПЗ якомога швидше.

ХР намагається зменшити ціну зміни вимог до ПЗ завдяки малим циклам розробки, а не одним довгим циклом. Екстремальне програмування сприймає зміни до вимог як звичайні, неминучі та бажані аспекти розробки ПЗ, і ці зміни мають бути очікуваним. Основна ідея полягає в тому що неможливо розробити самодостатній пакет вимог до ПЗ, зміни в вимогах – неминучі.

Екстремальне програмування також вводить набір практик та принципів на основі методології гнучкої розробки програмного забезпечення.

Екстремальне програмування описує чотири базові активності що виконуються при розробці програмного забезпечення: написання коду, тестування, слухання та дизайн. Написання коду. Прихильники ХР заявляють що єдиним дійсно важливим результатом розробки ПЗ є код: без готового коду нема продукту.

Тестування. Методологія екстремального програмування заявляє, що якщо дрібне тестування може перевірити незначну частину функціональності, то багато дрібних тестів можуть перевірити набагато більше частинок і продукт в цілому.

Основні прийоми ХР. Дванадцять основних прийомів екстремального програмування (за першим виданням книги Extreme programming explained) можуть бути об'єднані в чотири групи:

1. Короткий цикл зворотного зв'язку (Fine scale feedback).
 - 1.1. Розробка через тестування (Test driven development).
 - 1.2 Гра в планування (Planning game).
 - 1.3. Замовник завжди поруч (Whole team, Onsite customer).

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		50

- 1.4 Парне програмування (Pair programming).
- 2. Безперервний, а не пакетний процес.
 - 2.1 Безперервна інтеграція (Continuous Integration).
 - 2.2 Рефакторинг (Design Improvement, Refactor).
 - 2.3 Часті невеликі релізи (Small Releases).
- 3. Розуміння, що поділяється всіма учасниками.
 - 3.1 Простота (Simple design).
 - 3.2 Метафора системи (System metaphor).
 - 3.3 Колективне володіння кодом (Collective code ownership) або обраними шаблонами проектування (Collective patterns ownership).
 - 3.4 Стандарт кодування (Coding standard or Coding conventions).
- 4. Соціальна захищеність програміста (Programmer welfare), а саме 40 годинний робочий тиждень (Sustainable pace, Forty hour week).

Jira – була використана комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі-системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» – Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту – Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після – Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі – завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		51

бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад – Фотографіями, скріншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між станами можливі, визначається через настроюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

Завдяки універсальному підходу можна пристосувати Jira для багатьох непрофільних завдань, наприклад, керування вимогами, керування ризиками, аж до реалізації невеликої системи бронювання, автоматизації процесу рекрутингу.

Для інтеграції з зовнішніми системами підтримує інтерфейси SOAP, XML-RPC і REST. Поставляється із засобами інтеграції з такими системами управління версіями, як Subversion, CVS, Git, Clearcase, Team Foundation Server, Mercurial і Perforce.

Існують доповнення, що дозволяють вбудувати Jira в інтегровані середовища розробки, в тому числі Eclipse і IntelliJ IDEA. Перекладена багатьма мовами, включаючи українську, англійську, японську, німецьку, французьку, іспанську.

Для сторонніх розробників надаються кошти розробки розширень системи – плагінів. Розробники розширень можуть викладати плагіни для продажу на спеціальний розділ сайту Atlassian.

Є комерційним продуктом, який може бути ліцензований для роботи на локальному сервері або доступний в якості віддаленого додатки. Ціноутворення залежить від максимального числа користувачів, при цьому близько \$50 за

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		52

користувача для локального і \$7 на місяць за користувача для віддаленого доступу є типовими цінами.

Для академічних і комерційних клієнтів доступний повний вихідний код під ліцензією розробника.

Для проектів з відкритим вихідним кодом Atlassian надає спеціальну безкоштовну ліцензію при дотриманні наступних правил:

- проект використовує ліцензії, схвалені Open Source Initiative;
- Вихідний код проекту доступний для скачування;
- у проекту є публічно доступна веб-сайт;
- програмне забезпечення від Atlassian є на веб-сайті проекту.

Розглянемо протокол SNMP (Simple Network Management Protocol, простий протокол керування мережею) – це протокол керування мережами зв'язку на основі архітектури TCP/IP.

На основі концепції TMN в 1980–1990 р. різними органами стандартизації був вироблений ряд протоколів керування мережами передачі даних з різним спектром реалізації функцій TMN. До одного з типів таких протоколів керування належить Simple Network Management Protocol (SNMP).

SNMP – це технологія, покликана забезпечити керування й контроль за пристроями й програмами в мережі зв'язку шляхом обміну керуючою інформацією між агентами, що розташовуються на мережних пристроях, і менеджерами, розташованими на станціях керування. SNMP визначає мережу як сукупність мережних керуючих станцій й елементів мережі (головні машини, шлюзи й маршрутизатори, термінальні сервери), які спільно забезпечують адміністративні зв'язки між мережними керуючими станціями й мережними агентами. SNMP різних версій присвячений цілий ряд рекомендацій IETF (RFC).

Зазвичай при використанні SNMP присутні керовані та керуючі системи. До складу керованої системи входить компонент, який називається агентом, який відправляє звіти керуючій системі. По суті SNMP агенти передають управлінську

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		53

інформацію на керуючі системи як змінні (такі як «вільна пам'ять», «ім'я системи», «кількість працюючих процесів» тощо).

Керуюча система може отримати достовірну інформацію через операції протоколу GET, GETNEXT і GETBULK. Агент може самостійно без запиту надсилати дані, використовуючи операцію протоколу TRAP або INFORM. Управляючі системи можуть також відправляти конфігураційні оновлення або контролюючі запити, використовуючи операцію SET для безпосереднього управління системою. Операції конфігурування та управління використовуються тільки тоді, коли потрібні зміни у мережній інфраструктурі. Операції моніторингу зазвичай виконуються на регулярній основі.

Змінні, доступні через SNMP, організовані в ієрархії. Ці ієрархії та інші метадані (такі як тип і опис змінної) описуються Базами Керуючої Інформації (Management Information Bases (MIBs)).

SNMP не визначає, яку інформацію (які змінні) керована система повинна надавати. Навпаки, SNMP використовує розширювану модель, в якій доступна інформація визначається Базами Керуючої Інформації (MIB – Management Information Base).

Бази Керуючої Інформації описують структуру керуючої інформації пристроїв. Вони використовують ієрархічний адресний простір імен, що містить унікальний ідентифікатор об'єкта (object identifier (OID)).

Грубо кажучи, кожен унікальний ідентифікатор об'єкта ідентифікує змінну, яка може бути прочитана чи встановлена через SNMP. MIB'и використовують нотацію, визначену в ASN.1.

Ієрархія MIB може бути зображена як дерево з безіменним коренем, рівні якого приписані різними організаціями. На найвищому рівні MIB OID'и належать різним організаціям, що займаються стандартизацією, в той час як на нижчих рівнях OID'и виділяються асоційованими організаціями. Ця модель забезпечує управління на всіх шарах мережної моделі OSI, адже MIB'и можуть бути визначені для будь-яких типів даних і операцій.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		54

Керований об'єкт – це одна з будь-якого числа характеристик, специфічних для керованого пристрою. Керований об'єкт включає в себе один або більше екземплярів об'єкта (що ідентифікуються за OID), які насправді є змінними.

Існує два типи керованих об'єктів: Скалярні об'єкти визначають єдиний екземпляр об'єкта; Табличні об'єкти визначають множинні, пов'язані екземпляри об'єктів які групуються в таблицях MIB.

Прикладом керованого об'єкта може бути atInput, який є скалярним об'єктом що містить єдиний екземпляр об'єкта, ціле число, яке показує загальну кількість вхідних пакетів AppleTalk на мережний інтерфейс маршрутизатора.

Ідентифікатор об'єкта (OID) унікально ідентифікує керований об'єкт в ієрархії MIB.

В телекомунікаціях і комп'ютерних мережах, ASN.1 є стандартною гнучкою нотацією для опису структур даних, що служать для кодування, передачі і декодування даних. ASN.1 являє собою набір правил для опису структури об'єктів, незалежних від специфічних для обладнання методик кодування, і формальну нотацію, яка дозволяє уникнути неоднозначностей. ASN.1 це єдиний ISO та ITU-T стандарт, спочатку визначений у 1984 році як частина стандарту CCITT X.409: 1984. Пізніше, в 1988 році, завдяки його широкому застосуванню, він був виділений в окремий стандарт X.208. Значно переглянута версія 1995року описана в X.680. Адаптована підмножина ASN.1 Структура Керуючої Інформації (SMI) описана в протоколі SNMP для визначення наборів пов'язаних MIB об'єктів, також званих MIB модулями. SNMP працює на прикладному рівні TCP/IP (сьомий рівень моделі OSI).

Агент SNMP отримує запити по UDP-порту 161. Менеджер може посилати запити з будь-якого доступного порту джерела на порт агента. Відповідь агента буде відправлений назад на порт джерела на менеджері. Менеджер отримує повідомлення (Traps і InformRequests) по порту 162. Агент може генерувати повідомлення з будь-якого доступного порту. При використанні TLS або DTLS запити виходять по порту 10161, а пастки відправляються на порт 10162.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		55

У SNMPv1 зазначено п'ять основних протокольних одиниць обміну (protocol data units – PDU). Ще дві PDU, GetBulkRequest і InformRequest, були введені в SNMPv2 і перенесені в SNMPv3.

Нижче перераховані сім протокольних одиниць обміну SNMP:

1. GetRequest. Запит від менеджера до об'єкту для отримання значення змінної або списку змінних. Необхідні змінні вказуються в полі variable bindings (розділ поля values при цьому не використовується). Отримання значень зазначеної змінної повинно бути виконано агентом як Атомарна операція. Менеджеру буде повернений Response (відповідь) з поточними значеннями.

2. SetRequest. Запит від менеджера до об'єкту для зміни змінної або списку змінних. Зв'язані змінні вказуються в тілі запиту. Зміни всіх зазначених змінних повинні бути виконані агентом як атомарна операція. Менеджеру буде повернений Response з (поточними) новими значеннями змінних.

3. GetNextRequest. Запит від менеджера до об'єкту для виявлення доступних змінних і їх значень. Менеджеру буде повернений Response зі зв'язаними змінними для змінної, яка є наступною в базі MIB в лексикографічному порядку. Обхід всієї бази MIB агента може бути проведений ітераційним використанням GetNextRequest, починаючи з OID 0. Рядки таблиці можуть бути прочитані, якщо вказати в запиті OID-и колонок в пов'язаних змінних.

4. GetBulkRequest. Покращена версія GetNextRequest. Запит від менеджера до об'єкту для численних ітерацій GetNextRequest. Менеджеру буде повернений Response з декількома пов'язаними змінними, обійденими починаючи зі пов'язаної змінної (змінних) в запиті. Специфічні для PDU поля non-repeaters і max-repetitions використовуються для контролю за поведінкою відповіді. GetBulkRequest був введений в SNMPv2.

5. Response. Повертає зв'язані змінні і значення від агента менеджера для GetRequest, SetRequest, GetNextRequest, GetBulkRequest і InformRequest. Повідомлення про помилки забезпечуються полями статусу помилки і індексу помилки.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		56

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-1, який заснований на перетвореннях алгоритму SHA-1. SHACAL-1 шифрує 160-бітний блок даних з використанням 512-бітного ключа шифрування. Допускається використання більше коротких ключів шифрування (не коротше 128 біт), які перед виконанням розширення ключа (процедура розширення ключа також успадкована від SHA-1 і буде описана нижче) повинні бути доповнені нульовими бітами для досягнення 512-бітного розміру.

В алгоритмі SHACAL-1 передбачено 80 раундів шифрування. Шифруєме повідомлення представляється у вигляді п'яти 32-бітних субблоків A , B , C , D і E , над якими в кожному раунді виконуються наступні дії:

$$A_{i+1} = K_i + (A_i \lll 5) + f_i(B_i, C_i, D_i) + E_i + M_i,$$

$$B_{i+1} = A_i,$$

$$C_{i+1} = B_i \lll 30,$$

$$D_{i+1} = C_i,$$

$$E_{i+1} = D_i,$$

де i – номер раунду ($i = 0 \dots 79$),

K_i – фрагмент розширеного ключа для i -го раунду,

f_i – функція для i -го раунду (див. нижче),

$\lll$ – операція побітового циклічного зрушення вліво,

M_i – константи, що модифікують, певні в такий спосіб:

Раунди	Значення константи
0...19	5A827999
20...39	6ED9EBA1
40...59	8F1BBCDC
60...79	CA62C1D6

Використовувані в раундах функції f_i визначені так:

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		57

Раунди	Функція
0...19	$f(x,y,z)=(x\&y) (x'\&z)$
20...39,60...79	$f(x,y,z)=x \square \oplus y \square \oplus z$
40...59	$f(x,y,z)=(x \square \oplus y) (x \oplus z) (y \oplus z)$

У таблиці символами $\&$, $|$ і $\oplus$ позначені, відповідно, побітові логічні операції «і», «або» й «або, що виключає» (XOR); x' позначає побітовий комплемент до x .

Шифртекстом є конкатенація вмісту змінних A_{80} , B_{80} , C_{80} , D_{80} і E_{80} .

Процедура розширення ключа в алгоритмі SHACAL-1 також досить проста, вона виконується у два етапи:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0 \dots K_{15} \dots$

Етап 2. Інші фрагменти розширеного ключа $K_{16} \dots K_{79}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = (K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}) \lll 1.$$

Раунди розшифрування виконуються у зворотній послідовності; кожний з них має на увазі виконання наступних операцій:

$$A_i = B_{i+1},$$

$$B_i = C_{i+1} \lll 2,$$

$$C_i = D_{i+1},$$

$$D_i = E_{i+1},$$

$$E_i = K'_i + (B_{i+1} \lll 5)' + f'_i(C_{i+1} \lll 2, D_{i+1}, E_{i+1}) + A_{i+1} + M'_i + 4.$$

Тут запис $f(x)$ позначає побітовий комплемент результату виконання операції $f(x)$.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

- Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші: Налаштування; Довідка; Журнал роботи.
- Верхнього меню: Файл; Налаштування; Довідка.
- Розділу обрання групи налаштувань опцій роботи системи.
- Розділу виведення результату роботи системи у вигляді графіку.

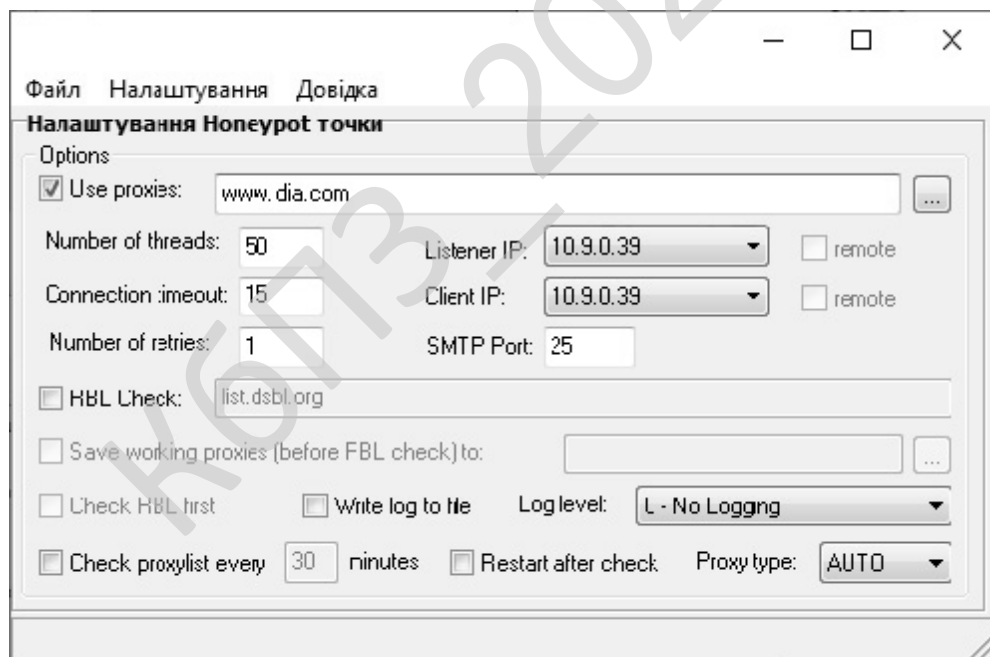


Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним

середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

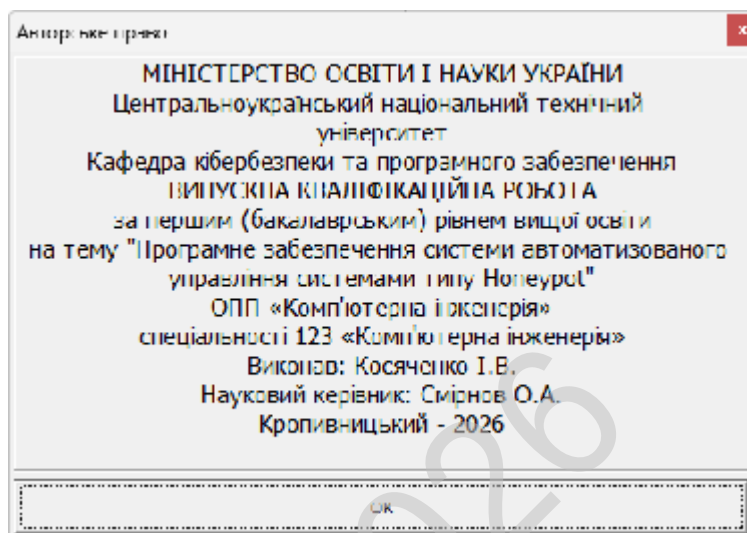


Рисунок 5.2 – Авторське право

Процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення. Таким чином у результаті вищерозглянутого можна стверджувати що розроблено інтерфейс системи у відповідності з вибраною метою роботи. Система містить максимальний необхідний набір функцій придатних для виконання будь-яких дій для забезпечення повноцінної роботи програми. Далі розглянемо висновки та використані літературні джерела.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		60

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи автоматизованого управління системами типу Honeyrot.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

- Був проведений огляд існуючих систем автоматизованого управління системами типу Honeyrot.
- Досліджена система автоматизованого управління системами типу Honeyrot.
- На основі отриманих результатів досліджень створена програмна реалізація системи автоматизованого управління системами типу Honeyrot.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання автоматизованого управління системами типу Honeyrot.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створенні програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи автоматизованого управління системами типу Honeyrot. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		61

загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-1.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

КБПЗ-2026

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		62

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.
2. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.
3. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.
4. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.
5. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.
6. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p.
7. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.
8. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.
9. Kuznetsov O., Frontoni E., Arnesano M., Smirnov O., Khruskov B. «Biometric Authentication in TinyML: Opportunities and Challenges». *Tiny Machine Learning: Design Principles and Applications*, 2026, pp. 587-633.
10. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing*, 2025. pp. 21-61.
11. Chulinda L., Smirnov O., Shapenko L., Ustynova I., Bohatiuk I., Kelyp S. «The role of innovation in ensuring the safety of international civil aviation». *Ceur Workshop Proceedings*, 2025, 4024, pp. 530–542.
12. Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед та ін. «Системи інформаційної зброї та технології інформаційної війни»: підручник / Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед, Жарков Я.М., Смірнов О.А,

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		63

Буравченко К.О., Давидюк А.В., Кононович В.Г., Корчинский В.В., Кудирко В.М., Фесенко А.О.; за заг. ред. В.М. Петрика, М.М. Присяжнюка.— К.: Видавничий центр “Кафедра”, 2025.— 320 с.

13. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка*. 2025. Том 1 № 29. С.704–716, 2025

14. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

15. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

16. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

17. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

18. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings*, 2024, 3909, pp. 227–241.

19. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

20. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

21. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2024. № 2(26), С. 170–188.

22. Смірнова Т.В., Коноплицька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка*. 2024. №4(24), С. 6-27.

23. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка*. 2024. №3(23), С. 111-131.

24. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем ІР-телефонії». *Підводні технології*, 2024, № 13, с. 28-35.

25. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

26. Akhalaia, G., Iavich, M., Iashvili, G., Prysiashnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings*, 2023, 3550, pp. 313-320.

27. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

28. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchев, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings*, Volume 3530, 2023, pp. 256-265.

29. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings*, Volume 3504, 2023, pp. 1-11.

30. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebesko, B. «Selection of a Rational Composition of Information Protection Means Using a Genetic Algorithm». In: *Rajakumar, G., Du, KL., Vuppapapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

31. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об'єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №3(19), 2023, С. 176-196.

32. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи розвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

33. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція “Інформаційна безпека та комп'ютерні технології” до 30-ти річчя*

кафедри кібербезпеки та програмного забезпечення, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

34. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». VII міжнародна науково-практична конференція “Інформаційна безпека та комп’ютерні технології” до 30-ти річчя кафедри кібербезпеки та програмного забезпечення, м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

35. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп’ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв’язку, 2023, вип. 2(72), С. 170-178.

36. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings*, Volume 3187, 2022,

37. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

38. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління*, № 2(70). 2022. С. 28-37.

39. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 3(69). С. 93-98.

					ВКРБ-123.26.0032.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		67

40. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»*, № 2 (307). С. 46-52. 2022.

41. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

42. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021*, Cracow, Poland, 22-25 September 2021. P. 414-418

43. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

44. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020*, Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

45. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings Volume 2805*, 2020, Pages 44-58.

46. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-

feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

47. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

48. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

49. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

50. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

51. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

52. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

53. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I., Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 166-171.