Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«14» червня 2022р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
бакалавра
(назва освітнього ступеня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Програмний засіб моніторингу журналу подій операційних систем у хмарному середовищі на базі ентропії»

Виконавець: студент IV курсу, групи КБ-41

_____ **Максим ПАНЧЕНКО**
(підпис) (ім'я прізвище)

	Прізвище, ініціали	**Підпис**
Керівник	Сергій ДАКОВ.	

Нормоконтроль	Юрій ЩЕБЛАНІН.	

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«01» листопада 2022 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студенту **КБ-41** **Панченку Максиму Валерійовичу**
(група) (прізвище ім'я по батькові)

Тема дипломної роботи Програмний засіб моніторингу журналу подій операційних систем у хмарному середовищі на базі ентропії

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №5 від 29.10.2021 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Програмне забезпечення для моніторингу безпеки хмарних обчислень на базі методу рухомого вікна для розрахунку ентропії

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Дослідження систем моніторингу журналів подій у хмарному середовищі, методи та засоби розробки програми моніторингу журналу подій на базі ентропії, розробка програми моніторингу журналу подій на базі ентропії, тестування розробленого програмного засобу.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність розроблено програмний засіб моніторингу журналу подій

операційних систем в хмарних середовищах на базі ентропії

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29.10.2021 року

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.10.2021 – 07.11.2022	*виконано*
2	Аналіз літератури та нормативно-правових документів	08.11.2022 – 20.11.2022	*виконано*
3	Розгляд архітектури програм моніторингу хмарних середовищ	21.11.2022 – 04.12.2022	*виконано*
4	Дослідження процесу розрахунку інформаційної ентропії	05.12.2022 – 14.01.2022	*виконано*
5	Розробка агента, що отримує записи журналу подій та розраховує ентропію	15.01.2022 – 27.02.2022	*виконано*
6	Розробка програми-аналізатора, що отримує дані від агента та виконує моніторинг	28.02.2022 – 10.04.2022	*виконано*
7	Тестування програмного засобу та моделювання деяких типів атак на інформаційну систему	11.04.2022 – 24.04.2022	*виконано*
8	Оформлення пояснювальної записки	25.04.2022 – 29.05.2022	*виконано*
9	Підготовка до захисту	30.05.2022 – 13.06.2022	*виконано*

Завдання видав _____ Сергій ДАКОВ
(підпис) (ім’я, прізвище)

Завдання прийняв до виконання _____ Максим ПАНЧЕНКО
(підпис) (ім’я, прізвище)

Термін подання дипломної роботи до ЕК 6 червня 2022 року

РЕФЕРАТ

Дипломна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел, 2 додатків, має 71 сторінки основного тексту, 46 рисунків та 6 формул. Список використаних джерел містить 19 найменувань і займає 3 сторінки.

Об'єкт дослідження – це процес виявлення несанкціонованих дій шляхом дослідження журналів подій в операційних системах сімейства Windows.

Метою роботи є розробка програмного засобу моніторингу журналу подій операційних систем в хмарному середовищі на базі ентропії.

Предмет дослідження – це механізм моніторингу журналів подій операційних систем та обчислення ентропії.

Методи дослідження: аналіз, моделювання, експеримент, тестування.

У роботі проаналізовано нормативно-правову базу щодо безпеки хмарних обчислень, літературу щодо моніторингу інформаційних систем, методів виконання та виявлення вторгнень, а також механізмів побудови системи моніторингу.

Запропоновано використовувати метод виявлення аномалій на базі ентропії для моніторингу журналу подій операційних систем.

Розроблено програмний засіб, що призначений для моніторингу журналу подій операційних систем в хмарних середовищах на базі ентропії.

Практичне значення роботи полягає у розробці програми моніторингу журналу подій операційних систем в хмарних середовищах на базі ентропії.

Результати здійснених у дипломній роботі досліджень можуть бути використані для виконання моніторингу аномалій в журналах подій операційних систем сімейства Windows.

У якості майбутніх досліджень може бути розроблено агенти для інших операційних систем, додано можливісь моніторингу декількох систем одночасно.

Ключові слова: моніторинг, журнал подій, операційні системи, ентропія, хмарні середовища, хмарні обчислення.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ISO	–	International Organization for Standardization
IEC	–	International Electrotechnical Commission
NIST	–	National Institute of Standards and Technology
EUCS	–	European Union Cybersecurity Certification Scheme on Cloud Services
ENISA	–	European Union Agency for Cybersecurity
TCP	–	Transmission Control Protocol
HTTP	–	HyperText Transfer Protocol
API	–	Application Programming Interface
HTML	–	HyperText Markup Language
XML	–	eXtensible Markup Language
SQL	–	Structured Query Language
SMTP	–	Simple Mail Transfer Protocol
ID	–	Identifier
NTFS	–	New Technology File System

ЗМІСТ

ВСТУП

Сьогодні використання хмарних обчислень є досить популярним, адже таке рішення може забезпечити високу доступність, а також дозволяє зменшити витрати на обслуговування власної інформаційної системи. Хоча частина відповідальності за обслуговування покладається на провайдера хмарного сервісу, користувач також має приймати в цьому участь. Серед основних моделей хмарних послуг «інфраструктура як сервіс» потребує найбільшого залучення користувача в обслуговуванні, адже вона передбачає надання лише «заліза» зі встановленою операційною системою, яку обирає користувач. За такої умови обслуговування операційної системи повністю покладається на користувача, зокрема забезпечення інформаційної безпеки.

Заходи захисту від кібератак не можуть надавати стовідсоткову гарантію неможливості проникнення зловмисника в інформаційну систему. Якщо зловмисник отримав доступ до системи, то потрібно якомога швидше виявити такі дії та перервати доступ зловмисника до інформаційної системи, а також провести розслідування для того, щоб виправити прогалини в безпеці. Тому **актуальними** є програмні засоби, що призначені для виявлення несанкціонованих дій в інформаційній системі. Одним із методів виявлення таких дій є моніторинг журналів подій.

Метою роботи є розробка програмного засобу моніторингу журналу подій операційних систем в хмарному середовищі на базі ентропії.

Об'єкт дослідження – це процес виявлення несанкціонованих дій в інформаційній системі шляхом дослідження журналів подій в операційних системах сімейства Windows.

Предмет дослідження – це механізм моніторингу журналів подій операційних систем та обчислення ентропії.

Методи дослідження: аналіз, моделювання, експеримент, тестування.

Новизна полягає у тому, що використано метод виявлення аномалій на базі ентропії для моніторингу журналу подій операційних систем. Зазвичай цей метод використовується для аналізу мережевого трафіку, наприклад, виявлення DDoS атак.

Практична цінність кваліфікаційної роботи полягає в розробці програмного засобу моніторингу журналу подій операційних систем в хмарних середовищах на базі ентропії.

Основні результати доповідалися та обговорювалися на науково-практичній конференції «Проблеми експлуатації та захисту інформаційно-комунікаційних систем» (Київ, 2022). Основні положення дипломної роботи викладені в 2 наукових працях, серед яких: 1 стаття у наукових фахових виданнях України, 1 – у матеріалах наукових конференцій.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ СИСТЕМ МОНІТОРИНГУ ЖУРНАЛІВ ПОДІЙ У ХМАРНОМУ СЕРЕДОВИЩІ

1.1 Нормативно-правова база щодо безпеки хмарних обчислень

Аналіз нормативно-правової бази показав, що українське законодавство до останнього часу ніяк не регулювало використання хмарних сервісів. Верховна Рада України 17 лютого 2022 року прийняла Закон України «Про хмарні послуги», який регулює використання хмарних послуг для органів державної влади. Цей закон набуває чинності 16 вересня 2022 року. У статті 8 закону є перелік заходів, що має вжити надавач хмарних послуг для управління ризиками безпеки [1]. Серед елементів, що мають враховувати ці заходи, зокрема, є моніторинг.

Також якщо хмарні сервіси використовуються для надавання електронних довірчих послуг, тоді мають враховуватись вимоги, що затверджені у Постанові Кабінету Міністрів України «Про затвердження вимог у сфері електронних довірчих послуг та Порядку перевірки дотримання вимог законодавства у сфері електронних довірчих послуг». Ці вимоги містять обов'язки адміністратора безпеки та аудиту, що включають в себе реєстрацію та моніторинг подій [2].

Для хмарних обчислень також є багато міжнародних нормативних документів, зокрема стандарти «EUCS – Cloud services scheme», «NIST SP 800-144 – Guidelines on Security and Privacy in Public Cloud Computing», «ISO/IEC 17788:2014».

«EUCS – Cloud services scheme» – це проект стандарту, розроблений агентством Європейського Союзу з кібербезпеки (ENISA), що описує схему забезпечення рівнів гарантій безпеки хмарних сервісів. Зокрема, цей стандарт має перелік заходів операційної безпеки, що включає в себе пункт «логування та моніторинг – ідентифікація подій» [3, с. 107]. У цьому пункті визначено вимоги для надавача хмарних послуг, що забезпечують рівнів гарантій «базовий» (basic), «значний» (substantial) та «високий» (high). Для рівня «базовий» про ідентифіковані

події, що можуть вказувати на інциденти безпеки, має бути сповіщено відповідним підрозділам організації для оцінки та усунення наслідків. Крім того, для рівня гарантій «значний» моніторинг подій має бути автоматизованим.

У стандарті «NIST SP 800-144» зазначено, що необхідно проводити постійний моніторинг безпеки мереж, інформації та систем організації, а також реагувати, приймаючи, уникаючи, або пом’якшуючи ризики [4, с. 9]. Крім того, збір та аналіз даних про стан системи має виконуватись регулярно, наскільки це необхідно для управління безпекою та ризиками [4, с. 20].

Ще один стандарт хмарних обчислень – «ISO/IEC 17788:2014». У цьому стандарті наведено список багатосторонніх аспектів хмарних обчислень. Одним із аспектів є безпека, вимоги якої включають в себе виконання моніторингу [5].

1.2 Журнал подій Windows

Для виявлення інцидентів безпеки в інформаційних системах виконується моніторинг на трьох рівнях: фізичному, математичному та аналітики даних [6]. Користувач моделі хмарних послуг «інфраструктура як сервіс» буде залучений до моніторингу останніх двох рівнів. Для розробки програми моніторингу було обрано математичний рівень, який включає в себе велику кількість компонентів, зокрема журнали подій операційних систем [6].

Для проведення досліджень обрано журнал подій операційних систем сімейства Windows, адже це досить популярні операційні системи, а також вони піддаються найбільшого ризику несанкціонованих впливів [7].

Як відомо [8], служба ведення журналу подій Windows використовує дані, що зберігаються в ключі реєстру «EventLog», який, в свою чергу, вміщує підключі, в яких зберігаються журнали протоколювання роботи системи. Основні журнали операційної системи Windows – це записи прикладних, системних процесів та процесів безпеки (рис. 1.1, 1.2, 1.3). Прикладні процеси можуть протоколювати події, що з ними пов’язані, в стандартний журнал прикладних процесів, або в свій окремий журнал, який може бути створений під час встановлення програми.

Приложение Событий: 16 971

Уровень	Дата и время	Источник	Код события	Категория задачи
Сведения	18.01.2022 19:17:00	RestartManager	10000	Отсутствует
Сведения	18.01.2022 19:16:34	SecurityCenter	15	Отсутствует
Сведения	18.01.2022 19:16:08	SecurityCenter	15	Отсутствует
Сведения	18.01.2022 19:16:07	Security-SPP	16394	Отсутствует
Сведения	18.01.2022 19:16:06	SecurityCenter	15	Отсутствует
Сведения	18.01.2022 17:51:15	edgeupdate	0	Отсутствует
Сведения	18.01.2022 17:45:32	edgeupdate	0	Отсутствует
Сведения	18.01.2022 17:37:57	igccservice	0	Отсутствует
Сведения	18.01.2022 17:37:36	Security-SPP	16384	Отсутствует
Сведения	18.01.2022 17:37:26	SecurityCenter	15	Отсутствует
Сведения	18.01.2022 17:36:40	Security-SPP	1040	Отсутствует
Сведения	18.01.2022 17:36:40	Security-SPP	1040	Отсутствует
Сведения	18.01.2022 17:36:36	Security-SPP	16394	Отсутствует
Сведения	18.01.2022 17:34:52	ELANFPService	0	Отсутствует
Сведения	18.01.2022 17:34:52	ELANFPService	0	Отсутствует

Рисунок 1.1 – Журнал прикладних процесів

Система Событий: 12 676 (!) Есть новые события

Уровень	Дата и время	Источник	Код события	Категория задачи
Сведения	18.01.2022 20:00:43	Kernel-Power	105	(100)
Сведения	18.01.2022 19:59:39	Kernel-Power	105	(100)
Сведения	18.01.2022 19:29:31	Service Control Manager	7040	Отсутствует
Сведения	18.01.2022 19:29:22	Service Control Manager	7040	Отсутствует
Сведения	18.01.2022 19:17:27	Service Control Manager	7040	Отсутствует
Сведения	18.01.2022 19:16:50	Service Control Manager	7040	Отсутствует
Сведения	18.01.2022 19:16:49	Service Control Manager	7040	Отсутствует
Сведения	18.01.2022 19:16:24	Kernel-General	16	Отсутствует
Сведения	18.01.2022 19:16:24	Kernel-General	16	Отсутствует
Сведения	18.01.2022 19:16:24	Kernel-General	16	Отсутствует
Сведения	18.01.2022 19:16:24	Kernel-General	16	Отсутствует
Сведения	18.01.2022 19:16:23	Kernel-General	16	Отсутствует
Сведения	18.01.2022 19:16:23	Kernel-General	16	Отсутствует
Сведения	18.01.2022 19:16:23	Kernel-General	16	Отсутствует
Сведения	18.01.2022 19:16:23	Kernel-General	16	Отсутствует

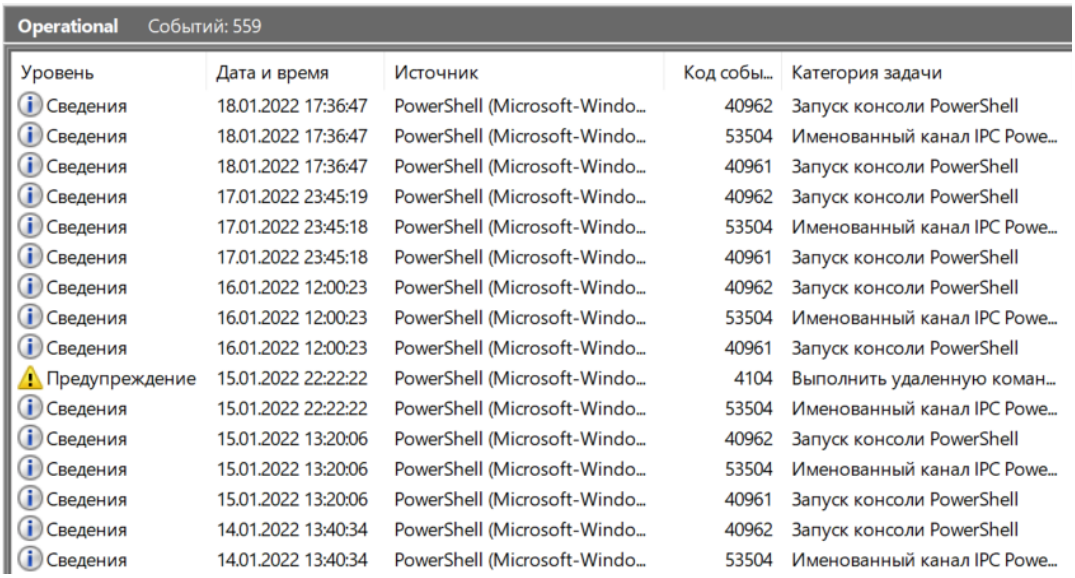
Рисунок 1.2 – Журнал системи

Безопасность Событий: 29 375 (!) Есть новые события

Ключевые слова	Дата и время	Источник	Код собы...	Категория задачи
Аудит успеха	18.01.2022 22:19:33	Microsoft Windows security auditing.	4672	Special Logon
Аудит успеха	18.01.2022 22:19:33	Microsoft Windows security auditing.	4624	Logon
Аудит успеха	18.01.2022 22:15:15	Microsoft Windows security auditing.	4672	Special Logon
Аудит успеха	18.01.2022 22:15:15	Microsoft Windows security auditing.	4624	Logon
Аудит успеха	18.01.2022 21:56:24	Microsoft Windows security auditing.	4672	Special Logon
Аудит успеха	18.01.2022 21:56:24	Microsoft Windows security auditing.	4624	Logon
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management
Аудит успеха	18.01.2022 21:55:43	Microsoft Windows security auditing.	5379	User Account Management

Рисунок 1.3 – Журнал безпеки

На основі детального аналізу структури та вмісту описаних журналів було прийнято рішення, що в рамках даного дослідження достатнім є використання журналу протоколювання роботи прикладних процесів, а також журналу протоколювання системи безпеки та PowerShell Operational, в якому зберігається інформація про події PowerShell, зокрема виконані в консолі команди (рис. 1.4).



Operational Событий: 559

Уровень	Дата и время	Источник	Код собы...	Категория задачи
Сведения	18.01.2022 17:36:47	PowerShell (Microsoft-Windo...	40962	Запуск консоли PowerShell
Сведения	18.01.2022 17:36:47	PowerShell (Microsoft-Windo...	53504	Именованный канал IPC Powe...
Сведения	18.01.2022 17:36:47	PowerShell (Microsoft-Windo...	40961	Запуск консоли PowerShell
Сведения	17.01.2022 23:45:19	PowerShell (Microsoft-Windo...	40962	Запуск консоли PowerShell
Сведения	17.01.2022 23:45:18	PowerShell (Microsoft-Windo...	53504	Именованный канал IPC Powe...
Сведения	17.01.2022 23:45:18	PowerShell (Microsoft-Windo...	40961	Запуск консоли PowerShell
Сведения	16.01.2022 12:00:23	PowerShell (Microsoft-Windo...	40962	Запуск консоли PowerShell
Сведения	16.01.2022 12:00:23	PowerShell (Microsoft-Windo...	53504	Именованный канал IPC Powe...
Сведения	16.01.2022 12:00:23	PowerShell (Microsoft-Windo...	40961	Запуск консоли PowerShell
Предупреждение	15.01.2022 22:22:22	PowerShell (Microsoft-Windo...	4104	Выполнить удаленную коман...
Сведения	15.01.2022 22:22:22	PowerShell (Microsoft-Windo...	53504	Именованный канал IPC Powe...
Сведения	15.01.2022 13:20:06	PowerShell (Microsoft-Windo...	40962	Запуск консоли PowerShell
Сведения	15.01.2022 13:20:06	PowerShell (Microsoft-Windo...	53504	Именованный канал IPC Powe...
Сведения	15.01.2022 13:20:06	PowerShell (Microsoft-Windo...	40961	Запуск консоли PowerShell
Сведения	14.01.2022 13:40:34	PowerShell (Microsoft-Windo...	40962	Запуск консоли PowerShell
Сведения	14.01.2022 13:40:34	PowerShell (Microsoft-Windo...	53504	Именованный канал IPC Powe...

Рисунок 1.4 – Журнал PowerShell Operational

1.3 Огляд існуючих програм моніторингу журналів подій

Існує багато систем моніторингу журналу подій, які дозволяють виявити несанкціоновані дії в інформаційній системі. Більшість із них ідентифікують такі дії шляхом аналізу вмісту записів у журналі подій на основі певних правил.

Прикладами програм моніторингу журналів подій є Splunk та Zabbix. Ці програми можуть опрацьовувати велику кількість даних, отримувати записи журналу із різних джерел, дозволяють виконувати пошук певних подій в журналах та створювати правила, на основі яких спрацьовують сповіщення, виконувати кореляцію, що дозволяє пов’язувати декілька подій, а також виконувати певні дії після спрацювання правил. Крім того, ці програми дозволяють створювати візуалізації подій у вигляді графіків, діаграм та таблиць.

Splunk дозволяє створювати правила на основі пошукових запитів. Таким чином можна зробити сповіщення про появу певних записів в журналі. Наприклад, про те, що кількість появи деякої події в журналі за одиницю часу перевищує певне встановлене значення. Крім того, Splunk дозволяє відправляти повідомлення на електронну пошту, а також встановлювати розширення з налаштованими сповіщеннями, наприклад, на основі машинного навчання, парсери для записів журналу із певних джерел, додаткові візуалізації, відправка повідомлень на інші сервіси, зокрема, Telegram.

Zabbix теж дозволяє зберігати та переглядати записи із різних журналів, створювати сповіщення про появу певних подій, надсилати повідомлення на пошту та інші сервіси, а також виконувати тести, наприклад, доступності сервісів.

Описані програми надають велику кількість інструментів для моніторингу журналів подій, але пошук інцидентів виконується саме за вмістом записів журналів. Якщо інцидент супроводжується записом великої кількості однакових подій, вміст яких не пов’язаний з помилками, або несанкціонованими діями, тоді він може бути не виявленим.

1.4 Постановка задачі

Враховуючи описані особливості роботи розглянутих систем моніторингу, задачею дипломного проектування обрано розробку програми, яка дозволяє виконувати моніторинг журналу подій на основі кількості різних повідомлень. Мірою такої кількості є інформаційна ентропія.

Таким чином, розроблена програма дозволить виявити аномалії, які спричиняють реєстрацію великої кількості будь-яких однакових або різних подій в журналі, навіть якщо вміст повідомлень може не вказувати на виконання несанкціонованих дій.

Програма може бути використана як доповнення до існуючих систем моніторингу за схемою, що описана у дослідженні [9]. Вона буде виконувати моніторинг аномалій та надсилати повідомлення про них адміністратору. Якщо

виявлені аномалії вказують на несанкціоновані дії, тоді можна сформувати правила або сигнатури для виявлення таких дій іншими системами моніторингу, наприклад, Splunk. Також розроблена програма може використовуватись як самостійна система моніторингу аномалій в журналах подій.

Для реалізації поставленого задання було виконано такі кроки:

1. Досліджено архітектуру програм моніторингу хмарних середовищ.
2. Досліджено процес розрахунку інформаційної ентропії для джерел, що безперервно генерують нові повідомлення.
3. Розроблено агент, що отримує записи журналу подій обраної операційної системи та розраховує ентропію.
4. Розроблено програму-аналізатор, що отримує дані від агента та виконує моніторинг.
5. Виконано тестування розробленого програмного засобу. Для цього змодельовано деякі типи атак на інформаційну систему.

Висновки за розділом 1

У першому розділі досліджено нормативно-правову базу щодо безпеки хмарних обчислень. Одним із важливих компонентів забезпечення безпеки таких систем є моніторинг, який виконується на трьох рівнях, зокрема математичному, який включає в себе журнали подій операційних систем.

Далі обрано операційну систему, на основі якої проведено дослідження. Також виконано аналіз журналу подій цієї операційної системи, для якого розроблено програму моніторингу. Крім того, розглянуто існуючі системи моніторингу журналу подій та визначено завдання дипломного проектування.

РОЗДІЛ 2
МЕТОДИ ТА ЗАСОБИ РОЗРОБКИ ПРОГРАМИ МОНІТОРИНГУ ЖУРНАЛУ ПОДІЙ НА БАЗІ ЕНТРОПІЇ

2.1 Модулі Python, використані під час створення програми

Програмний засіб розроблено на мові програмування Python. Ця мова має багато вбудованих модулів, а також додаткових модулів, які можна встановити з використанням пакетного менеджера «pip». Також цю мову програмування обрано через те, що вона забезпечує кросплатформеність, тобто розроблена програма може працювати на декількох операційних системах.

У програмі використано такі модулі Python:

- «tkinter» – вбудований модуль, що надає можливість створювати графічний інтерфейс. У програмі використано такі компоненти графічного інтерфейсу (віджети): «Tk» – основне вікно, після його закриття завершується виконання всієї програми; «Toplevel» – не основне вікно, його закриття не завершує виконання програми; «Frame» – контейнер для інших віджетів, представляє собою прямокутну область у вікні; «Label» – текстове поле; «Entry» – поле для вводу тексту; «Button» – кнопка; «PhotoImage» – картинка, яка зберігається в пам’яті, призначена для використання в інших віджетах; «messagebox» – вікно, що показує текстове повідомлення; «StringVar» – об’єкт, що зберігає текстові значення інших віджетів. Також використано додатковий модуль «ttk», який розширює список віджетів. Із цього модуля використано віджет «Spinbox», який призначений для вводу чисел та має стрілки, що збільшують та зменшують числове значення.
- «tkcalendar» – модуль, що додає до графічного інтерфейсу «tkinter» віджети «Calendar» та «DateEntry». У програмі використано віджет «DateEntry», який являє собою поле для вводу дати та надає можливість обрати дату в календарі.
- «matplotlib» – бібліотека, що надає можливість створювати графіки. У програмі використано такі модулі цієї бібліотеки: «pyplot» – включає в себе основні

функції для створення графіків, із цього модуля використано об’єкт «Figure», що являє собою картинку графіка; «style» – дозволяє обрати шаблон стилю графіка, тобто його зовнішній вигляд; «backends.backend_tkagg» – модуль, що дозволяє вбудувати графік у графічний інтерфейс «tkinter», зокрема містить об’єкт «FigureCanvasTkAgg», який являє собою віджет «tkinter», що містить графік; «ticker» – модуль, що дозволяє керувати позначками на графіку, із якого використано класи «AutoMinorLocator» та «MaxNLocator», які автоматично визначають позиції позначок; «dates» – модуль, що дозволяє робити графіки із датами; «animation» – модуль, що дозволяє робити анімацію графіків та містить об’єкт «FuncAnimation», що викликає функцію-обробник анімації зі вказаним часовим інтервалом.

- «socket» – вбудований модуль, що надає можливість взаємодіяти з мережевим інтерфейсом. У програмі з використанням цього модуля створюється TCP сервер і клієнт.
- «pickle» – вбудований модуль, що надає можливість комбінувати декілька об’єктів в один об’єкт «pickle», що представляє собою набір байтів. З використанням цього модуля виконується передача об’єкта «datetime» та значення ентропії від TCP сервера до клієнта.
- «threading» – вбудований модуль, що дозволяє виконувати код в окремому потоці.
- «io» – вбудований модуль, що дозволяє працювати з потоками вводу-виводу. Із цього модуля використано клас «BytesIO», що дозволяє зберігати набір байтів в пам’яті (буфері).
- «requests» – вбудований модуль, що дозволяє виконувати HTTP запити. Із використанням цього модуля виконується «POST» запит до HTTP API месенджера Telegram для відправки повідомлення через бот.
- «smtplib» – вбудований модуль, що дозволяє робити запити до поштового сервера за протоколом «SMTP». Відправка повідомлень на електронну пошту у розробленій програмі викоонується із використанням цього модуля.
- «ssl» – вбудований модуль, що дозволяє створювати захищені мережеві з’єднання. Із цього модуля використано функцію «create_default_context» для

з’єднання з поштовим сервером, яка призначена для створення контексту з налаштуваннями безпечного з’єднання за замовчуванням.

- «email» – вбудований модуль, що призначений для управління повідомленнями електронної пошти. Із цього модуля використано компонент «mime», який призначений для створення поштових повідомлень за стандартом «Multipurpose Internet Mail Extensions», що складаються з декількох частин. У програмі за допомогою цього компоненту створюється повідомлення електронної пошти, що містить текстову частину, HTML частину, а також зображення графіка.
- «queue» – вбудований модуль, що дозволяє створювати об’єкт-чергу. Цей об’єкт використано для передачі даних між потоками.
- «datetime» – вбудований модуль, що дозволяє створювати об’єкт, який містить даті і час.
- «sqlite3» – модуль, що надає інтерфейс для роботи з базою даних «SQLite».
- «ctypes» – бібліотека зовнішніх функцій. У програмі із використанням цього модуля викликається бібліотека Windows для активації масштабування графічного інтерфейсу програм. Для цього використана функція «ctypes.windll.shcore.SetProcessDpiAwareness».
- «os» – вбудований модуль, що призначений для взіємодії із операційною системою. У розробленій програмі він використовується для визначення операційної системи, на якій запущена програма. У залежності від операційної системи активується масштабування Windows та використовується різний код для встановлення іконки програми.
- «pathlib» – вбудований модуль, що призначений для роботи зі шляхами файлової системи. У розробленій програмі використовується для отримання шляху до папки «assets» із зображеннями для кнопок.
- «win32evtlog» – модуль, що є частиною пакету «pywin32», який надає можливість роботи із журналом подій операційної системи Windows. Із цього модуля використано функції «EvtSubscribe», що створює підписку на події, а також «EvtRender», яка дозволяє перетворити запис журналу в певний формат.

- «re» – вбудований модуль, що дозволяє працювати з регулярними виразами. У програмі використано функцію «search» для пошуку імені джерела повідомлень та коду події в отриманому XML записі про подію.
- «math» – вбудований модуль, що надає математичні функції. У програмі використано функцію «log2» для обчислення логарифму з основою 2.

2.2 Моделювання нормальної роботи системи

Для виявлення аномалій в роботі системи виконано порівняння еталонної ентропії системи з ентропією системи після виконання несанкціонованих дій.

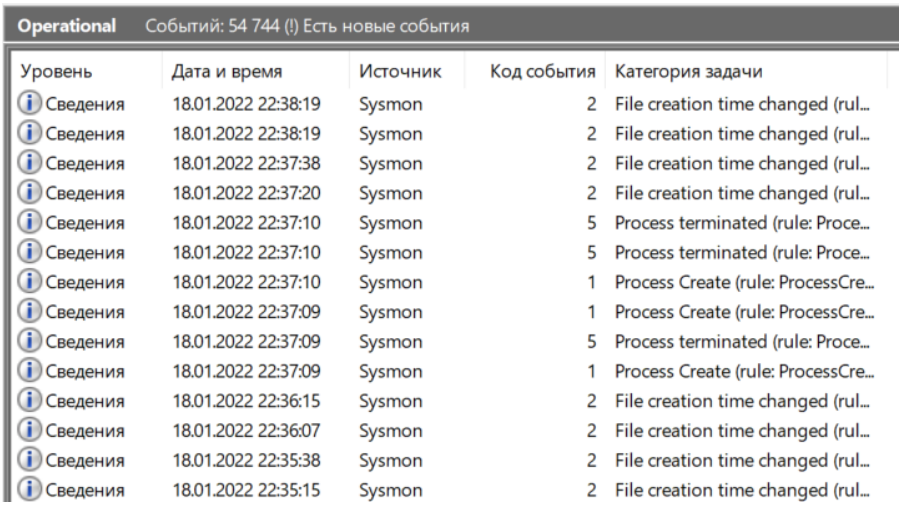
Вибірки для подальших досліджень сформовано на основі записів журналів подій при нормальній та аномальній роботі системи. Значення ентропії обчислено базуючись на імені джерела події (процесу, що записує подію в журнал).

Аналіз вмісту журналу подій показав, що за замовчуванням операційна система Windows не записує в журнал повідомлення про створення, зміну або видалення файлів, створення процесів, модифікацію реєстру. Але реєстрацію цих подій можна активувати, встановивши службу Sysmon.

Системний монітор (Sysmon) – це системна служба Windows, яка призначена для докладного протоколювання активності системи в журналі подій Windows. Ця служба надає детальну інформацію про запуск процесів, мережеві підключення, зміну часу створення файлів тощо [10].

Тому окрім стандартних записів в журналі протоколювання подій при нормальній роботі системи також виконано аналіз журналу із записами служби Sysmon.

Під час встановлення служби Sysmon використано стандартні конфігурації, що доступні для завантаження за посиланням [11]. На цьому етапі досліджень в роботі не виконувався аналіз мережевих підключень, тому записи типу «NetworkConnect» були вимкнені. Розрахунок ентропії для повідомлень Sysmon виконано на основі коду події. Фрагмент даних для аналізу представлений на рисунку 2.1.



Operational Событий: 54 744 (!) Есть новые события

Уровень	Дата и время	Источник	Код события	Категория задачи
Сведения	18.01.2022 22:38:19	Sysmon	2	File creation time changed (rul...
Сведения	18.01.2022 22:38:19	Sysmon	2	File creation time changed (rul...
Сведения	18.01.2022 22:37:38	Sysmon	2	File creation time changed (rul...
Сведения	18.01.2022 22:37:20	Sysmon	2	File creation time changed (rul...
Сведения	18.01.2022 22:37:10	Sysmon	5	Process terminated (rule: Proce...
Сведения	18.01.2022 22:37:10	Sysmon	5	Process terminated (rule: Proce...
Сведения	18.01.2022 22:37:10	Sysmon	1	Process Create (rule: ProcessCre...
Сведения	18.01.2022 22:37:09	Sysmon	1	Process Create (rule: ProcessCre...
Сведения	18.01.2022 22:37:09	Sysmon	5	Process terminated (rule: Proce...
Сведения	18.01.2022 22:37:09	Sysmon	1	Process Create (rule: ProcessCre...
Сведения	18.01.2022 22:36:15	Sysmon	2	File creation time changed (rul...
Сведения	18.01.2022 22:36:07	Sysmon	2	File creation time changed (rul...
Сведения	18.01.2022 22:35:38	Sysmon	2	File creation time changed (rul...
Сведения	18.01.2022 22:35:15	Sysmon	2	File creation time changed (rul...

Рисунок 2.1 – Журнал служби Sysmon

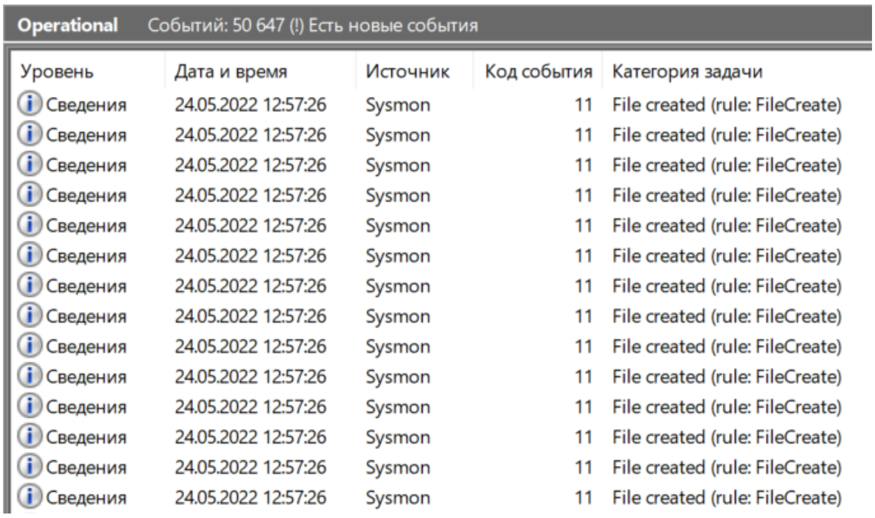
Для обчислення еталонних значень ентропії в системі виконувались прикладні процеси, що представлені такими додатками, як Opera, Microsoft Office 2019 та месенджером Telegram.

2.3 Моделювання несанкціонованих дій в системі

На наступному етапі досліджень виконано моделювання несанкціонованих дії в системі з активною службою Sysmon. У межах реалізації цього етапу було виконано запис великої кількості файлів в директорію «Завантаження». Служба Sysmon реєструвала в журналі подій створення файлів в цій директорії (рис. 2.2). За необхідності в конфігурацію служби можна додати будь-які інші директорії, для яких буде реєструватись зміна файлів. Також було створено велику кількість процесів, дії протоколювались з занесенням відповідних записів в журнал протоколювання подій службою Sysmon.

Створення великої кількості файлів або процесів може вказувати на локальні DoS атаки, що спрямовані на деградацію продуктивності програмно-апаратного комплексу, і, як наслідок, порушення доступності як сервісу ІБ. Крім того, зміна одночасно великої кількості файлів може свідчити про несанкціоновані дії, наприклад, роботу віруса-шифрувальника.

Також було виконано атаку, що спрямована на отримання несанкціонованого віддаленого доступу до системи за допомогою зворотного TCP з’єднання.



Operational   Событий: 50 647 (!) Есть новые события

Уровень	Дата и время	Источник	Код события	Категория задачи
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)
Сведения	24.05.2022 12:57:26	Sysmon	11	File created (rule: FileCreate)

Рисунок 2.2 – Повідомлення про створення файлів у журналі служби Sysmon

Для виконання цієї атаки використано віртуальну машину Linux із Metasploit Framework. Це платформа з відкритим вихідним кодом, яка підтримує дослідження вразливостей, розробку експлойтів і створення спеціальних інструментів безпеки [12]. Дана платформа вміщує велику базу експлойтів.

Дослідження літератури показало, що існують методи створення експлойта-оболонки, які дозволяють уникнути виявлення антивірусом [13]. Основуючись на припущенні, що зловмисник вже обійшов антивірус Microsoft Defender, перед виконанням атаки його було вимкнено.

Було створено виконуваний файл із корисним навантаженням «Meterpreter reverse TCP» за допомогою інструменту MSFvenom, що є частиною Metasploit. Цей виконуваний файл використано для встановлення з’єднання з цільовою системою. У консолі Metasploit обрано експлойт handler (обробник) із тим самим корисним навантаженням. Після запуску експлойта обробник очікує підключення на визначеному порті. Створений виконуваний файл скопійовано в каталог веб-сервера Apache та завантажено на цільовій системі. Після відкриття цей файл встановлює з’єднання з Metasploit, в результаті чого створюється сесія, що надає доступ до цільової системи. Доступ і, відповідно, процеси в системі виконуються від імені

того користувача, що відправив на виконання завантажений файл. Для отримання прав суперкористувача чи «Системи» потрібно виконати ескалацію привілеїв. Для цього використано експлойт «FodHelper UAC bypass», що дозволяє обійти контроль доступу користувачів Windows. Далі виконується команда «getsystem» в сесії Meterpreter, яка дозволить отримати доступ до системи від імені користувача «Система», що має найвищі привілеї. Схема виконаної атаки показана на рисунку 2.3.

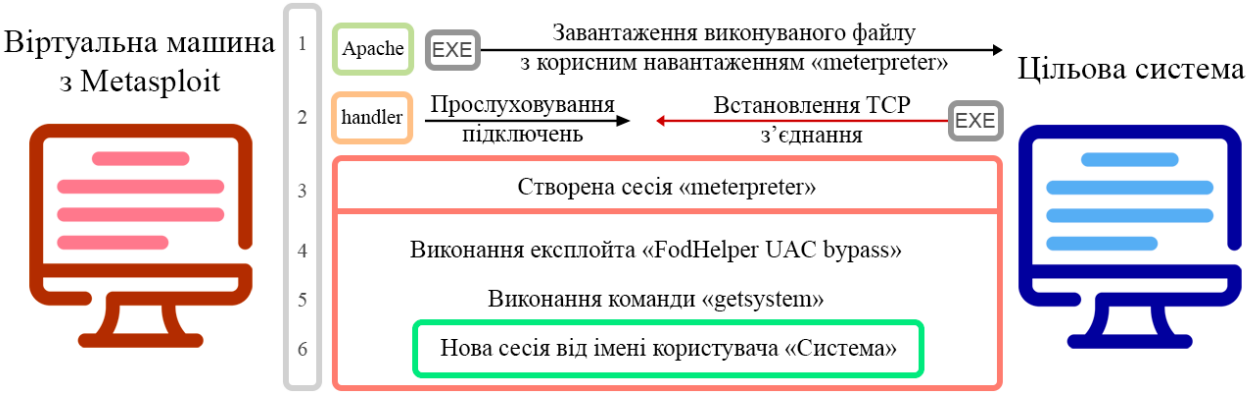


Рисунок 2.3 – Схема виконання атаки

Описані дії призводять до протоколювання подій в журналі, зокрема створення процесів, зміна реєстру, виконання команд PowerShell, що повинно впливати на значення ентропії.

2.4 Розрахунок ентропії журналу подій

Ентропію джерела повідомлень розраховано за формулою Шеннона [14, с. 32]:

$$H(A) = -\sum_{i=1}^{k} p_i \log_2 p_i, \tag{2.1}$$

де A – множина повідомлень, k – кількість повідомлень, p_i – імовірність появи кожного повідомлення.

Для розрахунку ймовірності появи повідомлення використано формулу:

$$p_i = \frac{n_i}{N},$$

де n_i – кількість повідомлень певного типу, N – загальна кількість повідомлень.

Для розрахунку ентропії журналу подій використано метод рухомого вікна [15]. Згідно цього методу ентропія розраховується для повідомлень у межах обраного вікна розміром W. Для перших W повідомлень ентропія розраховується за формулою (2.1), далі із появою нових повідомлень вікно зсувається на величину dW.

На рисунку 2.4 показано схему методу рухомого вікна зі значеннями $W = 5$ та $dW = 1$, різні повідомлення позначені цифрами.

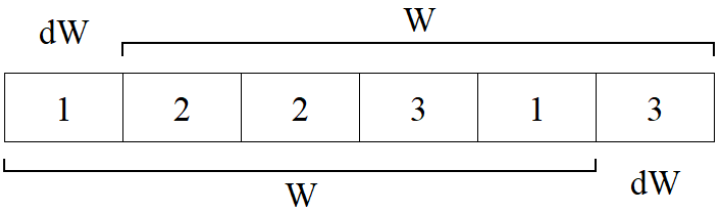


Рисунок 2.4 – Схема методу рухомого вікна

Після зсуву вікна розраховується поточне значення ентропії:

$$H_i = H_{i-1} + \Delta H.$$

Зміна ентропії розраховувалась за формулою:

$$\Delta H = H_i - H_{i-1}. \quad (2.2)$$

Підставивши формулу (2.1) у формулу (2.2) отримано вираз:

$$\Delta H = \left(-\sum_{i=1}^{k} p_i \log_2 p_i \right) - \left(-\sum_{j=1}^{l} {p'}_j \log_2 {p'}_j \right), \quad (2.3)$$

де літери без штриха – значення ймовірності повідомлень для поточного вікна, зі штрихом – для попереднього вікна.

Ті елементи, імовірність яких у поточному і попередньому вікні не змінилась, не будуть впливати на зміну ентропії, отже вона буде залежати лише від тих елементів, що надійшли та вибули з вікна. На рисунку 2.4 після зсуву вікна зміниться ймовірність елементів 1 і 3, а для елементу 2 залишиться незмінною. Якщо позначити попередні значення ймовірності елементів 1 і 3 як p'_1 та p'_3, а поточні – p_1 та p_3, тоді зміна ентропії буде розраховуватись за формулою:

$$\Delta H = (-p_1 \log_2 p_1 - p_3 \log_2 p_3) - (-p'_1 \log_2 p'_1 - p'_3 \log_2 p'_3).$$

Тобто, для елементів, значення ймовірності яких змінилось, потрібно додати ентропію після зсуву вікна та відняти ентропію до зсуву.

Висновки за розділом 2

У цьому розділі описано методи та засоби, які використано під час розробки програми моніторингу журналу подій. Програму розроблено на мові програмування Python, яка забезпечує кросплатформеність. Описано використані модулі цієї мови програмування, а також моделювання нормальної роботи системи та несанкціонованих дій. Крім того, описано розрахунок ентропії журналу подій методом рухомого вікна.

Під час обчислення ентропії окрім стандартних журналів операційної системи Windows враховано журнал служби Sysmon, яка протоколює додаткові події в системі, зокрема створення і зміну файлів, створення процесів і записів в реєстрі

тощо. Велика кількість таких записів в журналі може вказувати на несанкціоновані дії. Також описано процес моделювання атак на інформаційну систему.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМИ МОНІТОРИНГУ ЖУРНАЛУ ПОДІЙ НА БАЗІ ЕНТРОПІЇ

3.1 Архітектура розробленої програми

Архітектура систем моніторингу хмарних середовищ зазвичай включає в себе аналізатор даних та агент [16; 17]. Агент розміщується в хмарі та призначений для збору даних, які необхідні для виконання моніторингу. Зібрані дані надсилаються аналізатору, який, відповідно, виконує їх аналіз та демонструє результати моніторингу.

У розробленій програмі агент отримує записи журналу та розраховує ентропію. Значення ентропії надсилається аналізатору, який виконує перевірку того, що ці значення не перевищують встановлені користувачем допустимі значення, виводить графік зміни ентропії від кількості подій, а також записує отримані значення в базу даних. Якщо виявлено перевищення, тоді надсилаються відповідні повідомлення на електронну пошту та Telegram. Для надсилання значень ентропії встановлюється TCP з'єднання. Агент виступає в ролі сервера та прослуховує підключення від аналізатора. Схема архітектури програмного засобу показана на рисунку 3.1.

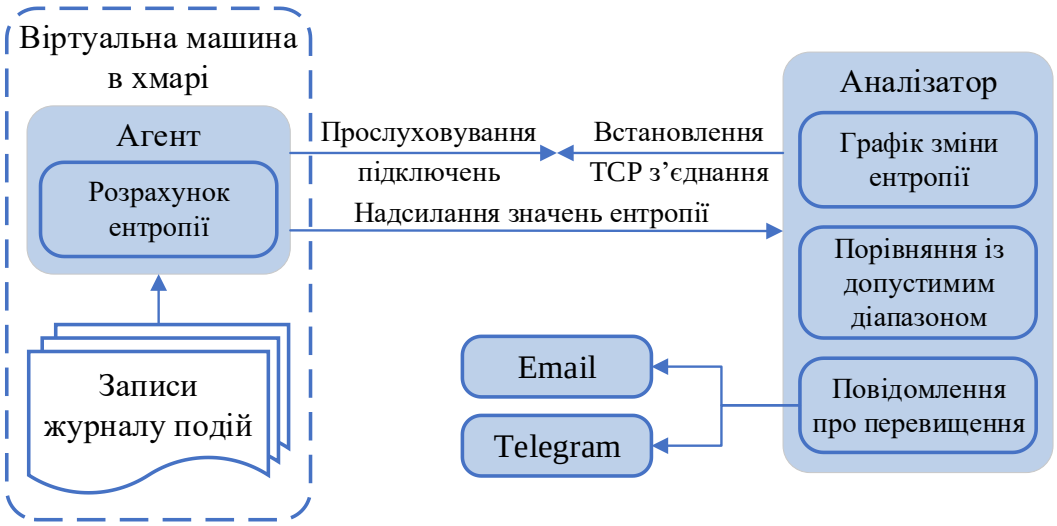


Рисунок 3.1 – Архітектура програми моніторингу журналу подій

3.2 Агент програми моніторингу журналу подій

Агент виконує два завдання – розрахунок ентропії журналу подій та надсилання результатів аналізатору. Для отримання записів журналу подій використано функцію «EvtSubscribe» модуля «win32evtlog», який є частиною пакета «pywin32», що включає в себе функції Win32 API. Схема роботи агента показана на рисунку 3.2.

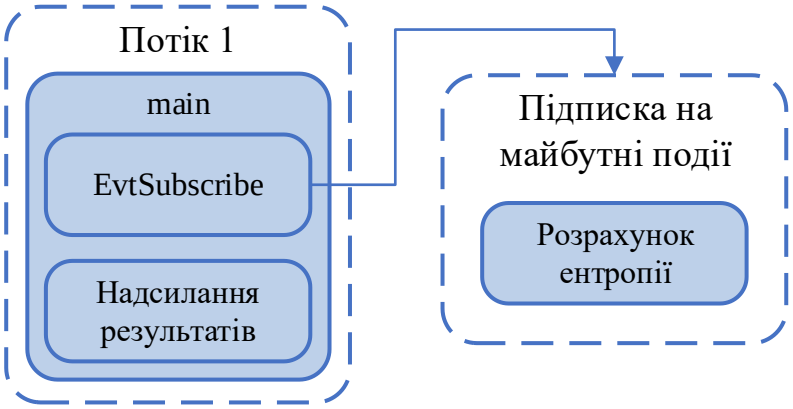


Рисунок 3.2 – Схема виклику основних функцій агента

Функція «EvtSubscribe» створює об’єкт, що вміщує інформацію про підписку на події обраного журналу. У якості параметрів функції вказано назву журналу, обрано майбутні події, за що відповідає параметр «EvtSubscribeToFutureEvents», а також вказано функцію-обробник «new_logs_event_handler», яка буде викликана після появи нових подій. Описана функція може приймати лише однин журнал, тому було створено список, до якого в циклі додаються об’єкти підписки для кожного окремого журналу. Коли з’являється нова подія, тоді в окремому потоці викликається функція «new_logs_event_handler», у якій за допомогою регулярних виразів із модуля «re» серед усієї інформації про подію вибирається назва джерела (процесу, що записав подію), або код події, далі викликається функція розрахунку ентропії.

Після створення підписок викликається функція «server», що призначена для надсилання результатів обчислення ентропії за допомогою TCP з’єднання. Для

створення TCP сервера використано модуль «socket». Він прослуховує підключення на порту 1234 та приймає підключення лише від одного клієнта. Після встановлення з'єднання відбувається отримання значення ентропії із об'єкта черги «Queue» за допомогою функції «get». Якщо черга порожня, тоді виконання потоку зупиниться доки не з'являться об'єкти в черзі. Після надсилання отриманих даних викликається функція об'єкта черги «task_done», що видаляє поточне значення ентропії із черги. Схема взаємодії потоків з використанням об'єкта черги показана на рисунку 3.3.

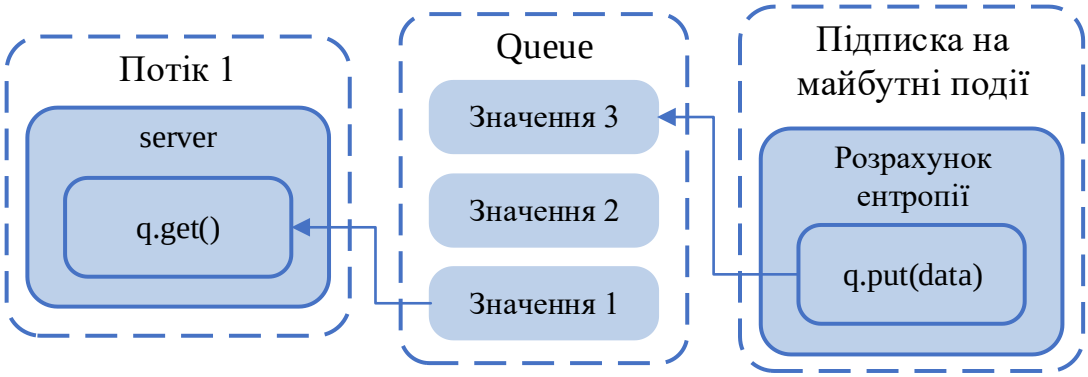


Рисунок 3.3 – Схема взаємодії потоків

Розрахунок ентропії методом рухомого вікна описано у функціях «calc», «calcH», «calcdH», «before», «now». Інформація для розрахунку ентропії зберігається в таких глобальних змінних:

- ціле число «w» – розмір вікна;
- ціле число «dw» – величина, що позначає розмір зсуву вікна;
- список «seq» – послідовність подій у вікні;
- словник «itemsn» – поточна кількість кожної події у вікні;
- словник «itemsf» – попередня частота кожної події у вікні;
- дійсне число «H» – значення ентропії;
- «lock» – об'єкт «Lock» із модуля «threading», який використовується для блокування одночасного доступу до змінних.
- «q» – об'єкт класу «Queue», що призначений для передачі адреси агента в потік відправки повідомлень.

Основна функція розрахунку ентропії – «calc». Вона викликається із функції «new_logs_event_handler» після отримання ідентифікатора події, яким є назва джерела (провайдера), або код події. Так як після появи події функція розрахунку викликається в окремому потоці, для унеможливлення одночасного виконання розрахунку ентропії для декількох подій, що спричинить помилку в розрахунках через одночасне використання змінних, які зберігають інформацію для розрахунку, використано об’єкт «Lock» із модуля «threading». На початку функції «calc» викликано функцію «acquire» цього об’єкту, що активує блокування коду для інших потоків та в кінці – функцію «release», яка знімає блокування. У функції створено додаткову змінну «n», яка позначає довжину списка «seq».

Після запуску агента ця функція додає події в список «seq» поки не заповниться вікно. Після заповнення вікна розраховується кількість кожної події у вікні та записується в словник «itemsn», де ідентифікатору події відповідає її кількість. Після появи нових подій буде розраховуватись кількість лише для тих подій, що знаходяться в межах вікна, а для тих, що вийшли з вікна кількість записується як 0. Ця умова реалізована перевіркою того, що індекс подій в списку «seq» більше за вираз «n-w-1», який буде відповідати індексу останньої події, яка вибула з вікна. Далі розраховуються ймовірності появи подій та записуються в словник «itemsf», у якому ідентифікатору події відповідає її частота. Словник «itemsf» буде використано під час розрахунку наступного значення ентропії як частоту подій до зсуву вікна. Після формування словника «itemsf» на основі розрахованих частот обчислюється перше значення ентропії. Розрахунок частот усіх подій у вікні та першого значення ентропії виконується лише один раз після заповнення вікна. Наступні дії виконуються тоді, коли надходять нові події та зсувається вікно. Якщо події, що надійшла, немає в списку «itemsf», тоді вона додається до цього списку зі значенням частоти 0. Коли кількість нових подій відповідає величині dW виконується розрахунок зміни ентропії, яка додається до попереднього значення ентропії. Після цього зі списку «seq» видаляються елементи, що вибули з вікна та отримане значення ентропії передається в об’єкт черги використовуючи функцію «put». Передане значення отримується функцією «socket»

та відправляється аналізатору. Схема роботи функції «calc» показана на рисунку 3.4. Для розрахунку ентропії із функції «calc» викликаються функції «calcH» та «calcdH».

Перша функція виконує розрахунок ентропії за формулою Шеннона (2.1), друга – розраховує зміну ентропії за формулою (2.3). Для розрахунку зміни ентропії в список «chseq» записуються ідентифікатори подій, які надійшли та вибули з вікна.

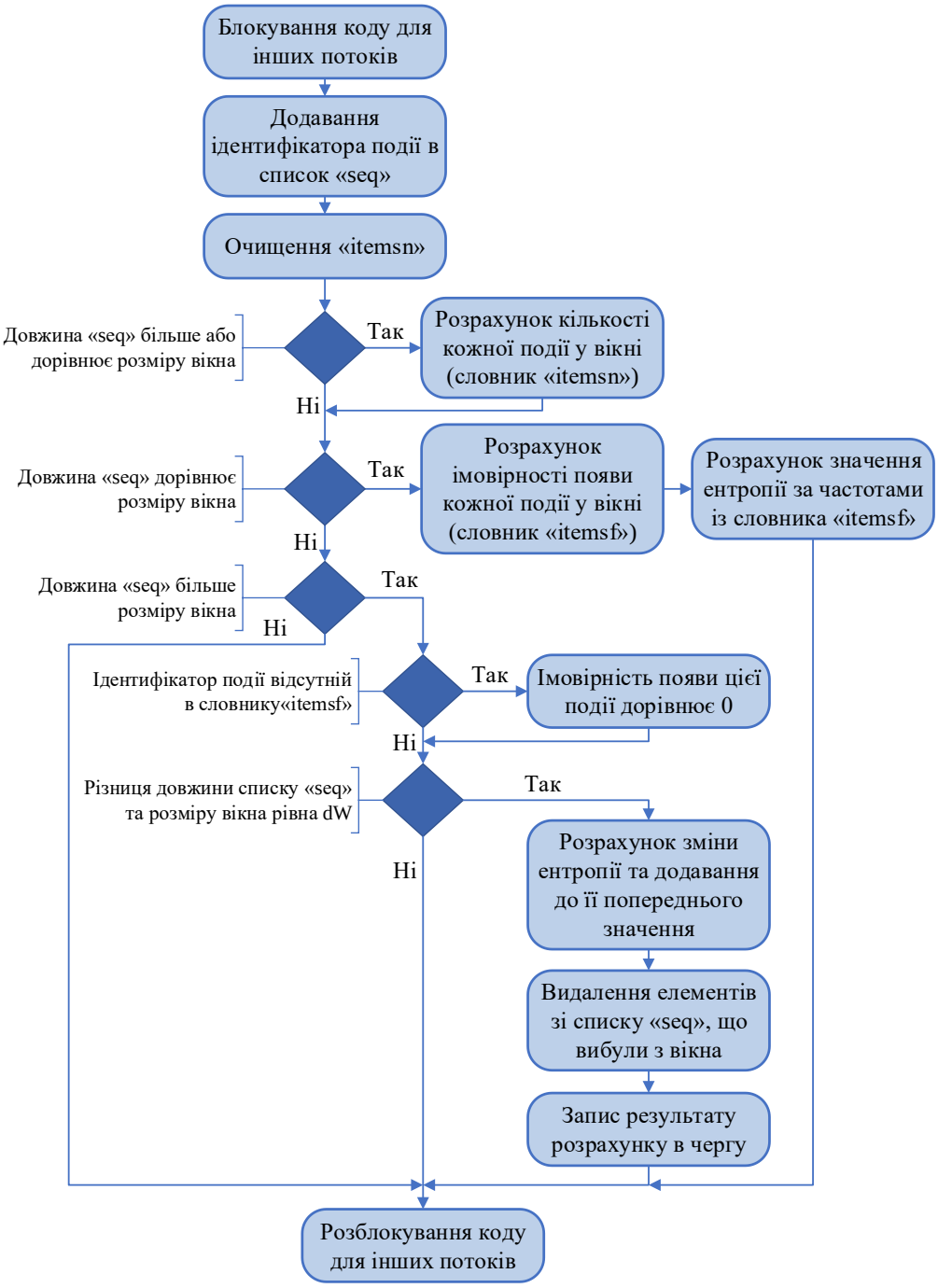


Рисунок 3.4 – Схема роботи функції «calc»

Далі виконується розрахунок ентропії для цих подій до зсуву з використанням даних зі словника «itemsf», що описано у функції «before», а також – після зсуву з використанням даних зі словника «itemsn», що описано у функції «now». Результат розрахунку функції «before» віднімається від результату, отриманого з функції «now». Отримана зміна ентропії повертається в функцію «calc», а також в словник «itemsf» записуються нові частоти подій для поточного стану вікна, використовуючи кількості появи подій у вікні зі словника «itemsn». Схема роботи функції «calcdH» показана на рисунку 3.5.

Рисунок 3.5 – Схема роботи функції «calcdH»

3.3 Аналізатор значень ентропії

Аналізатор розроблено з використанням об’єктно-орієнтованого програмування. У класах розміщений увесь код, що пов’язаний з графічним інтерфейсом, а також з потоками. Крім того, код різних функціональних частин програми розміщено в різних файлах:

- «Entropy_client.py» – основний файл, в якому створюється головне вікно;
- «main_window.py» – включає в себе класи, що описують складові графічного інтерфейсу головного вікна;
- «main_functions.py» – функції, що викликаються із головного вікна;

- «history_window.py» – класи, що описують складові графічного інтерфейсу вікна історії значень ентропії;
- «shared_gui.py» – класи, що описують графічний інтерфейс, який використовуються в головному вікні і у вікні історії
- «db_functions.py» – функції, що пов’язані з базою даних;
- «relative_to_assets.py» – функція, що повертає шлях до каталогу «assets», у якому розміщено графічні зображення для кнопок.

Схема зв’язків між цими файлами (імпортування) показана на рисунку 3.6.

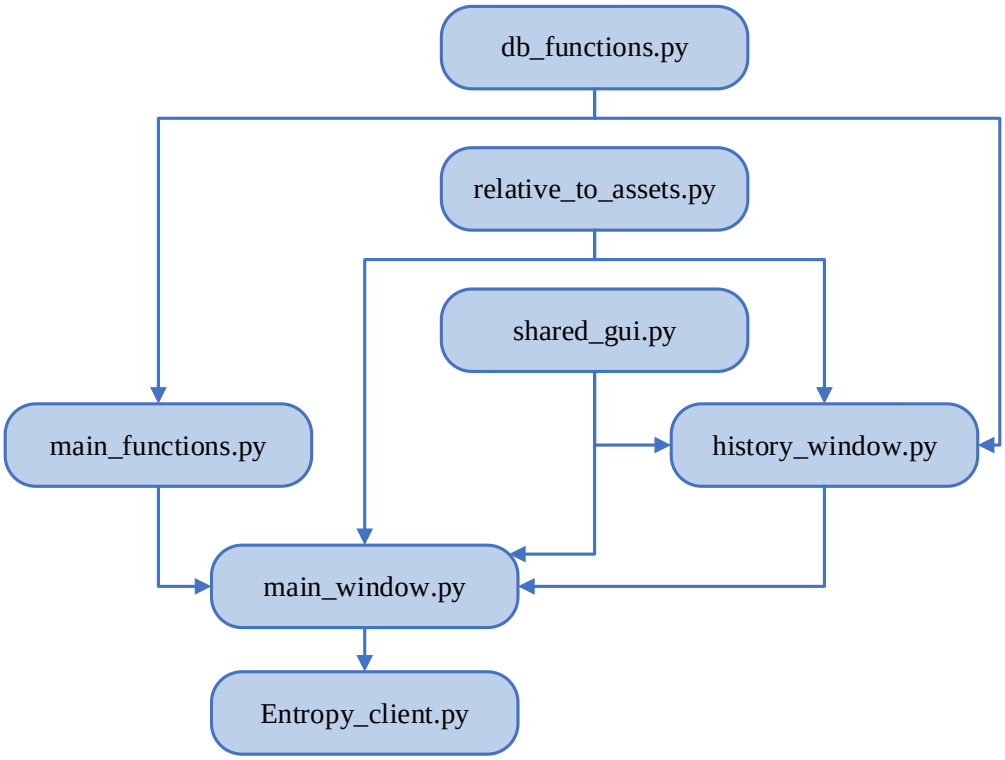


Рисунок 3.6 – Схема зв’язків між файлами програми

Програму розроблено з використанням трьох потоків. Перший потік виконує дії, що пов’язані з графічним інтерфейсом, другий потік – це TCP клієнт, який отримує значення ентропії від агента, третій потік призначений для відправлення повідомлень про перевищення на електронну пошту та Telegram. Функції потоків описані в класах «Tcp_client_thread» та «Send_message_thread».

Використання окремого потоку для TCP клієнта засноване на тому, що для роботи графічного інтерфейсу необхідне постійне виконання коду для обробки

натискань кнопок, пересування вікна, тощо. Але сокет, який використовується для встановлення підключення, є блокуючим, тобто інший код не може виконуватись доки не завершиться операція з сокетом. Також відправка повідомлень може спричиняти затримки, тому ці дії теж виокремлено в окремому потоці.

На рисунку 3.7 показано схему роботи аналізатора та зв'язки між функціями, класами, змінними та файлами.

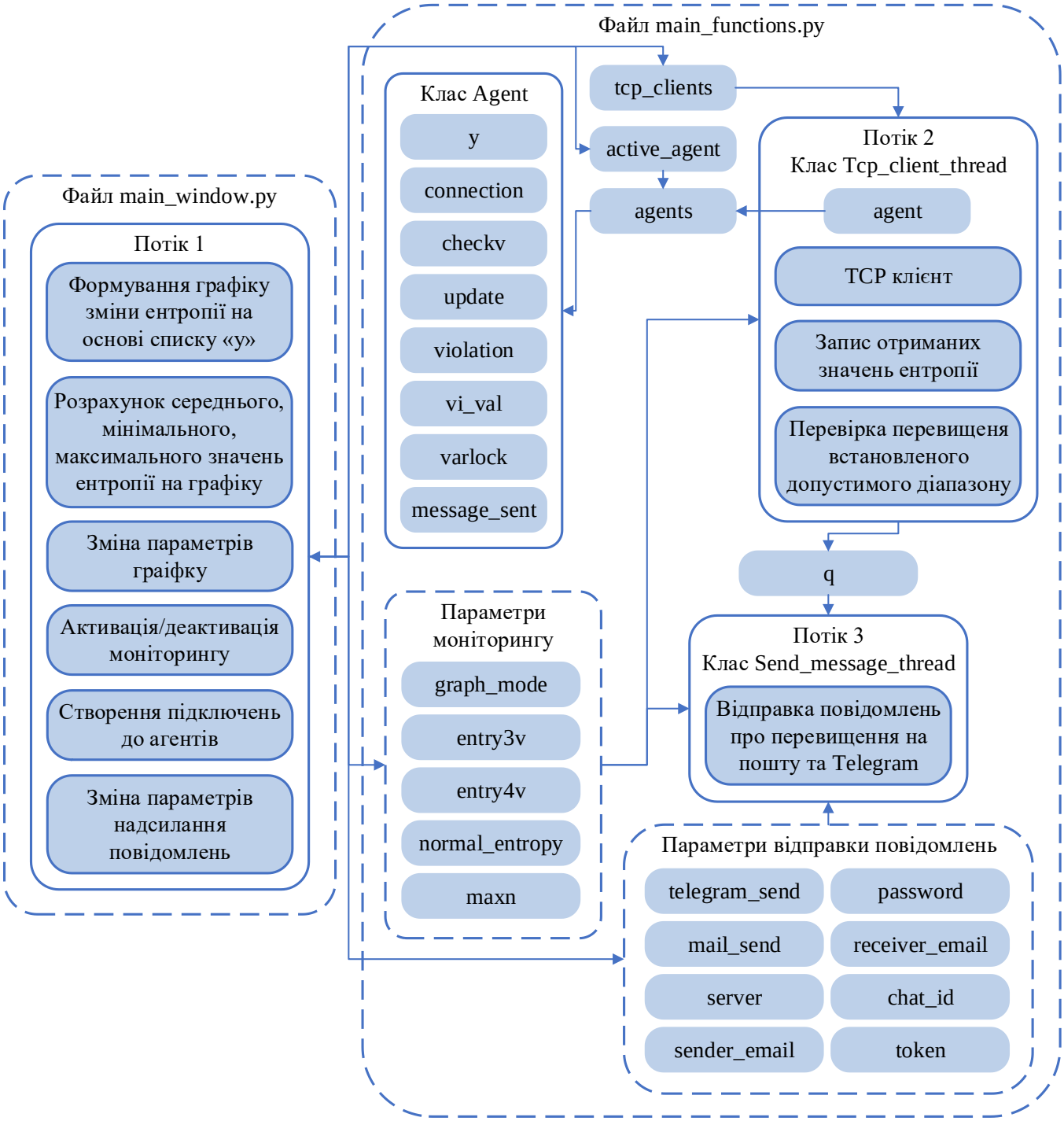


Рисунок 3.7 – Схема роботи аналізатора та взаємодії потоків

Передача інформації між потоками виконується з використанням змінних, що оголошені в файлі «main_functions.py». Змінні, що пов’язані з агентом зберігаються в об’єктах класу «Agent»:

- список «y» – значення ентропії, що відображаються на графіку;
- булева змінна «connection» – присутнє підключення до агента;
- булева змінна «checkv» – активований моніторинг;
- булева змінна «update» – виконання оновлення графіку;
- булева змінна «violation» – перевищення дозволеного діапазону значень ентропії;
- число «vi_val» – значення ентропії, що перевищує встановлений діапазон;
- булева змінна «message_sent» – повідомлення відправлено;
- «varlock» – об’єкт «Lock» із модуля «threading», який використовується для блокування одночасного доступу до змінних.

Крім того, у файлі «main_functions.py» оголошено словники «agents» та «tcp_clients», у які записуються об’єкти класу «Agent» та «Tcp_client_thread». Також у змінній «active_agent» зберігається адреса поточного агента.

Також інформація, що пов’язана з налаштуваннями моніторингу і відправки повідомлень, зберігається в глобальних змінних файлу «main_functions.py»:

- булева змінна «graph_mode» – режим графіку;
- «entry3v» – верхня допустима межа ентропії або значення ентропії за нормальної роботи (в залежності від режиму графіку);
- «entry4v» – нижня допустима межа ентропії або допустиме відхилення(в залежності від режиму графіку);
- булева змінна «normal_entropy» – значення ентропії за нормальної роботи в режимі графіка «відхилення ентропії» (в режимі графіка «зміна ентропії» ця змінна рівна нулю);
- ціле число «maxn» – максимальна кількість подій на графіку;
- булева змінна «telegram_send» – відправка повідомлень в Telegram;
- булева змінна «mail_send» – відправка повідомлень на пошту;

- рядок «server» – адреса агента;
- рядок «sender_email» – поштова адреса відправника повідомлення;
- рядок «password» – пароль для поштової адреси відправника;
- рядок «receiver_email» – поштова адреса отримувача повідомлення;
- рядок «chat_id» – ідентифікатор користувача Telegram;
- рядок «token» – API токен боту Telegram, з якого будуть відправлятись повідомлення;
- «q» – об’єкт класу «Queue», що призначений для передачі адреси агента в потік відправки повідомлень.

3.4 Класи і функції аналізатора

Файл, що призначений для запуску програми – «Entropy_client.py». У цьому файлі описана функція «main», що виконується після запуску програми. У цій функції створюється головне вікно «root» та створюється об’єкт класу «main_gui» із файлу «main_window.py», який описує всі елементи графічного інтерфейсу головного вікна. Графічний інтерфейс у програмі розроблено з використанням модуля «tkinter». Також у файлі описано функцію «on_closing», яка буде виконана після закриття головного вікна. На рисунку 3.8 показано схему зв’язків між описаними функціями.

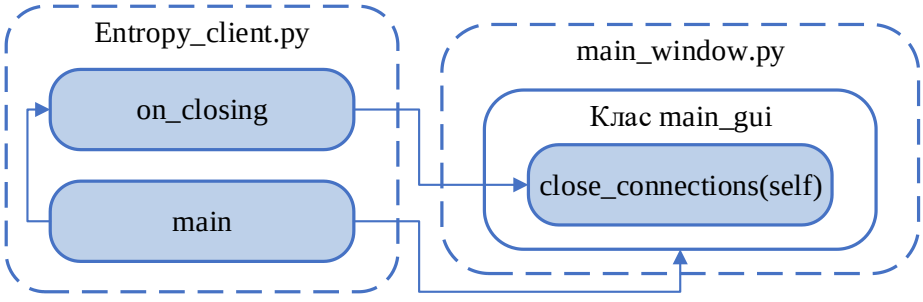


Рисунок 3.8 – Функції файлу «Entropy_client.py»

У файлі «main_window.py» описано класи, що формують графічний інтерфейс головного вікна. У функції «__init__» класу «main_gui» створюються об’єкти класів, які описують окремі елементи графічного інтерфейсу:

- «Graph_gui» – клас, що описує анімацію графіку, на якому відображається зміна ентропії. Цей клас наслідує клас «Graph_gui» із файлу «shared_gui.py», в якому описано основні функції, що відображають графік. Для графіку використано модуль «matplotlib».
- «Graph_type» – клас, що відображає панель, на якій пропонується обрати тип графіку – «Зміна ентропії» або «Відхилення ентропії». У цьому класі доповнено функції, що викликаються під час натискання кнопок, які відповідають типу графіку. Наслідує клас «Graph_type» із файлу «shared_gui.py», у якому описано основні елементи панелі.
- «Graph_params» – клас, що відображає панель, на якій встановлюються параметри графіку. У цьому класі описуються елементи панелі та створюється об’єкт класу «Graph_type». Із цього класу викликається функція «change_events_on_graph» у файлі «main_functions.py».
- «Main_colorboxes» – клас, у якому на вікні створюються кольорові блоки з використанням об’єкта «canvas» та назва вікна.
- «Connect_gui» – клас, що відображає панель, яка призначена для встановлення підключення до агента та відкриття вікна історії значень ентропії. Із цього класу викликається функція «add_agent» або «connect» у файлі «main_functions.py» для встановлення з’єднання із агентом, а також створюється об’єкт класу «History_window» із файлу «history_window.py».
- «Message_gui» – клас, що відображає панель, на якій встановлюються параметри відправлення повідомлень на електронну пошту та Telegram.

На рисунку 3.9 показано зв’язки між класами та функціями головного вікна.

Основна функціональна частина головного вікна – графік зміни ентропії, який описаний в класах «Graph_gui» із файлів «main_window.py» та «shared_gui.py».

У файлі «shared_gui.py» описана основна структура класу «Graph_gui», яка використовується у головному вікні і вікні історії значень ентропії. Клас містить такі функції:

- «__init__» – ініціалізація класу. У цій функції створюються основні компоненти графіка – фрейм, малюнок графіка, осі графіка, а також налаштування

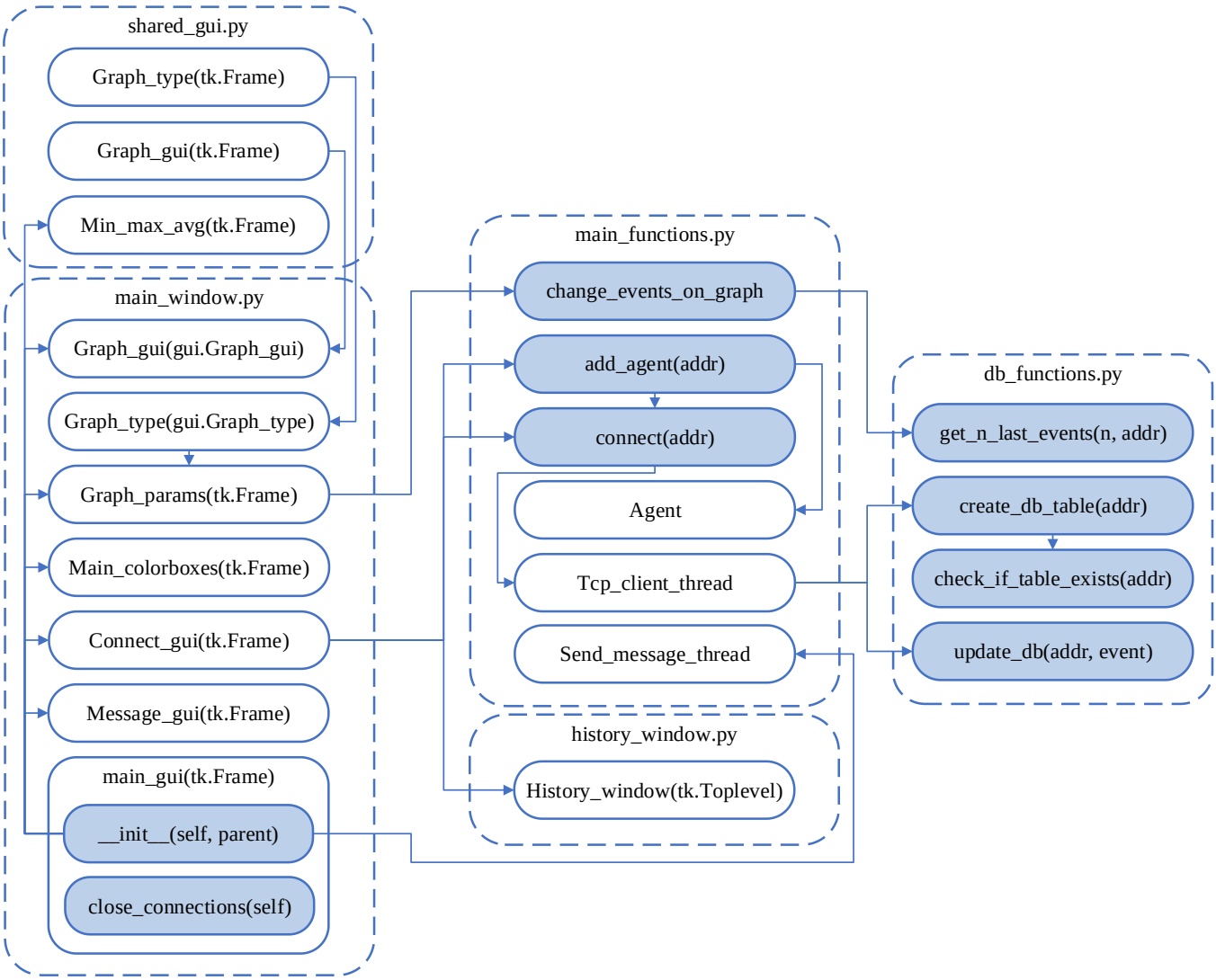


Рисунок 3.9 – Класи та функції головного вікна

компонентів графіка. Для встановлення параметрів осей графіка викликається функція «set_axis_params».

- «clear_graph» – функція, що очищає дані осей графіка.
- «set_axis_params» – функція, що встановлює параметри осей – для встановлення параметрів осі «y» викликається функція «set_y_axis_params», для осі «x» встановлюється мінімальне значення «0» і цілі значення поділок.

- «set_y_axis_params» – функція, що встановлює параметри осі «y». Встановлюється мінімальне значення «0», а також розмір порожнього місця між графіком та верхньою частиною області побудови.

- «plot_graph» – функція, що виконує побудову графіка на основі отриманого списку значень осі «y». Спочатку викликається функція «clear_graph», далі розраховуються значення для осі «x», що відповідають кількості елементів списку «y», за допомогою функції «range». Далі зафарбовується простір під графіком та виконується побудова графіка за списками «x» та «y». Після цього викликається функція «set_axis_params».

- «show_min_max_avg» – функція, що встановлює значення текстових полів на панелі, яка описана в класі «Min_max_avg» та показує середнє, мінімальне і максимальне значення ентропії на графіку. Ця функція отримує змінну «target», в якій записується об’єкт класу «Min_max_avg», для якого змінюється текст, а також список «y», що містить значення ентропії на графіку.

У файлі «main_window.py» описано клас «Graph_gui», що наслідує клас із файлу «shared_gui.py». Цей клас містить такі функції:

- «__init__» – ініціалізація класу. У цій функції створюється об’єкт анімації графіка «animation.FuncAnimation», у параметрах якого вказано фрейм із графіком, функцію, яка виконує анімацію – «animate», та інтервал в мілісекундах, із яким буде викликатись функція анімації.

- «animate» – функція, що виконує анімацію. Функція приймає обов’язковий параметр, що містить кількість повторів анімації, якому присвоєно назву «i». У цій функції описаний параметр не використовується. Спочатку у функції виконується перевірка того, чи не порожня змінна «active_agent» у файлі «main_functions.py». Якщо змінна не порожня, тоді це означає, що підключення до агента виконано. Далі об’єкту класу «Agent», якому відповідає елемент словника «agents» із ключем, що записаний в змінній «active_agent», надається ім’я «agent». Після цього використовується конструкція «with agent.varlock», що виконує записаний далі блок коду із використанням об’єкта «Lock». Ця конструкція необхідна для того, щоб унеможливити одночасний доступ до змінних класу «Agent» із різних потоків.

Конструкція «with agent.varlock» ідентична записам «agent.varlock.acquire()» «agent.varlock.relase()» на початку та в кінці блоку коду. Далі до змінної agent.y присвоюється ім’я «y» та за умови, що змінна «agent.update» має значення «True» і довжина списку «y» більша за 1, викликаються функції «plot_graph» та «show_min_max_avg», яким передаються відповідні параметри. В кінці змінній «agent.update» присвоюється значення «False».

На рисунку 3.10 показано схему зв’язків між описаними класами, функціями та змінними.

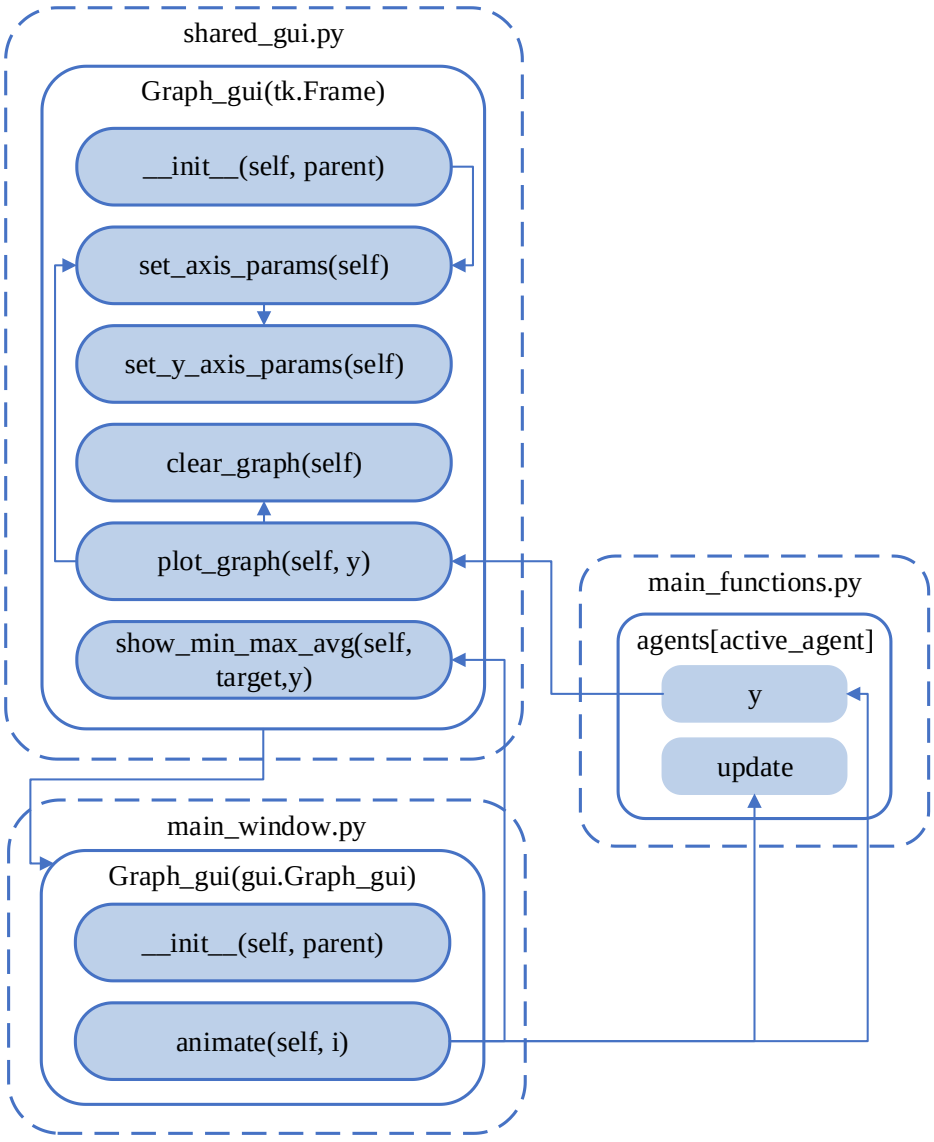


Рисунок 3.10 – Схема роботи графіка зміни ентропії на головному вікні

Для встановлення параметрів графіка використовується панель, що описана в класах «Graph_type» у файлах «main_window.py» та «shared_gui.py», а також у класі «Graph_params» у файлі «main_window.py».

Клас «Graph_type» описує панель, у якій пропонується обрати тип графіку. У файлі «shared_gui.py» клас «Graph_type» містить такі функції:

- «__init__» – ініціалізація класу. У цій функції створюються текстові поля, кнопки, що призначені для зміни типу графіку, а також булева змінна «button_graph_mode», яка позначає обрану кнопку: значення «True» – натиснута перша кнопка, значення «False» – друга кнопка.
- «radio_button1» – функція, що викликається після натискання першої кнопки. У цій функції встановлюється значення «True» для змінної «button_graph_mode», а також змінюється картинка обох кнопок, що позначає їх обрання.
- «radio_button2» – функція, що викликається після натискання другої кнопки. У цій функції встановлюється значення «False» для змінної «button_graph_mode» та змінюється картинка обох кнопок.

У файлі «main_window.py» клас «Graph_type» наслідує однойменний клас із файлу «shared_gui.py». Цей клас містить такі функції:

- «radio_button1» – доповнює функцію натискання першої кнопки. У цій функції змінюється значення текстових полей на панелі, що писана в класі «Graph_params», а також видаляються попередні значення в об’єктах «entry», що призначені для введення параметрів моніторингу.
- «radio_button2» – доповнює функцію натискання другої кнопки. У цій функції також змінюється значення текстових полей та видаляються попередні значення в об’єктах «entry».

Клас «Graph_params» у файлі «main_window.py» описує панель, що призначена для введення параметрів графіку. Цей клас містить такі функції:

- «__init__» – ініціалізація класу. У цій функції текстові поля на панелі, об’єкти «entry», об’єкт класу «Graph_type», а також кнопки, що призначені для

підтвердження введених параметрів моніторингу та увімкнення/вимкнення моніторингу.

- «apply_graph_settings» – функція, що виконується після натискання кнопки «Підтвердити». Спочатку виконується перевірка, що об’єкти «entry» не порожні, далі код записано в конструкції «try-except», яка призначена для перехоплення виключень. Виконується конвертування тексту із об’єктів «entry» в числовий тип. Якщо замість числа користувач записав текст, то буде згенеровано виключення та виведено відповідне повідомлення. Далі виконується перевірка, що кількість подій на графіку більша за 5, інакше виводиться відповідне повідомлення. Після виконання перевірок отримані числові значення записуються в змінні «entry3v», «entry4v», «graph_mode», «normal_entropy», «maxn» у файлі «main_functions.py». Після цього викликається функція «change_events_on_graph» із файлу «main_functions.py», яка призначена для оновлення списку «y», що містить значення ентропії, на основі яких будується графік. Спочатку виконується перевірка, що змінна «active_agent» не порожня. Далі об’єкту класу «Agent», якому відповідає елемент словника «agents» із ключем, що записаний в змінній «active_agent», надається ім’я «agent». Після цього використовується конструкція «with agent.varlock». У цій конструкції виконується очищення списку «y» об’єкта «agent» та викликається функція «get_n_last_events» із файлу «db_functions.py». Ця функція отримує кількість подій та адресу агента і виконує SQL запит до бази даних для отримання вказаної кількості останніх значень ентропії відповідного агента та повертає отриманий список. Елементи цього списку перебираються в циклі «for» та записуються в список «y». У кінці змінній «update» об’єкта «agent» присвоюється значення «True». Останній блок коду функції «apply_graph_settings» видаляє попереднє значення об’єкта «entry», що містить максимальну кількість подій та записує значення зі змінної «maxn» у файлі «main_functions.py».
- «monitor_button» – функція, що призначена для увімкнення та вимкнення моніторингу, викликається після натискання відповідної кнопки. Спочатку виконується перевірка, що змінна «active_agent» не порожня. Далі об’єкту класу «Agent», якому відповідає елемент словника «agents» із ключем, що записаний в

змінній «active_agent», надається ім’я «agent». Після цього виконуються перевірки, що усі введені параметри моніторингу підтверджені, тобто значення об’єктів «entry» збігається зі значенням відповідних змінних, та виводяться відповідні повідомлення. Далі якщо моніторинг не виконувався, то він вмикається – змінюється картинка і текст кнопки, якій відповідає дана функція, а також встановлюється значення «True» для змінної «agent.checkv» і «False» для змінної «agent.message_sent». Якщо моніторинг виконувався і не було виявлено перевищення, то моніторинг вимикається – змінюється картинка і текст кнопки, а також встановлюється значення «False» для змінної «agent.checkv». Якщо моніторинг виконувався і було виявлено перевищення, то моніторинг відновлюється – змінюється картинка і текст кнопки, а також встановлюється значення «True» для змінної «agent.checkv» і «False» для змінних «agent.message_sent» і «agent.violation». На рисунку 3.10 показано схему зв’язків між описаними класами, функціями та змінними.

Клас «Connect_gui» описує панель, що призначена для підключення агента. Цей клас містить такі функції:

- «__init__» – ініціалізація класу. У цій функції створюється панель (об’єкт «frame») і об’єкт «entry», що призначені для вводу адреси, а також текстове поле і кнопки «Підключитись» та «Історія». Також створюється змінна «history_opened» зі значенням «False», яка позначає відкриття вікна історії значень ентропії.

- «button_1_handler» – функція, що викликається після натискання кнопки «Підключитись». Спочатку виконується перевірка того, що вказаної адреси агента немає серед ключів словника «agents» із файлу «main_functions.py». Далі введена адреса агента записується в змінну «active_agent», а також викликається функція «add_agent» із файлу «main_functions.py», якій передається адреса агента. Ця функція створює об’єкт класу «Agent» в словнику «agents», у якості ключа використовується адреса агента. Далі викликається функція «connect», яка створює об’єкт класу «Tcp_client_thread», запускає потік цього об’єкта, а також очищає список «y» і встановлює значення «True» для змінної «connection». Якщо словник «agents» вже містить вказану адресу агента і змінна «connect» має значення «False», тоді викликається функція «connect» і передається адреса агента. Якщо об’єкт

«entry», що призначений для вводу адреси агента, порожній, тоді виводиться відповідне повідомлення.

- «history_button» – функція, що викликається після натискання кнопки «Історія». Ця функція створює об’єкт класу «History_window» із файлу «history_window.py», якщо змінна «history_opened» має значення «False». Також для змінної «history_opened» встановлюється значення «True». Якщо ця змінна має значення «True», тоді вікно значень ентропії переводиться на передній план.

Схема описаного класу показана на рисунку 3.11.

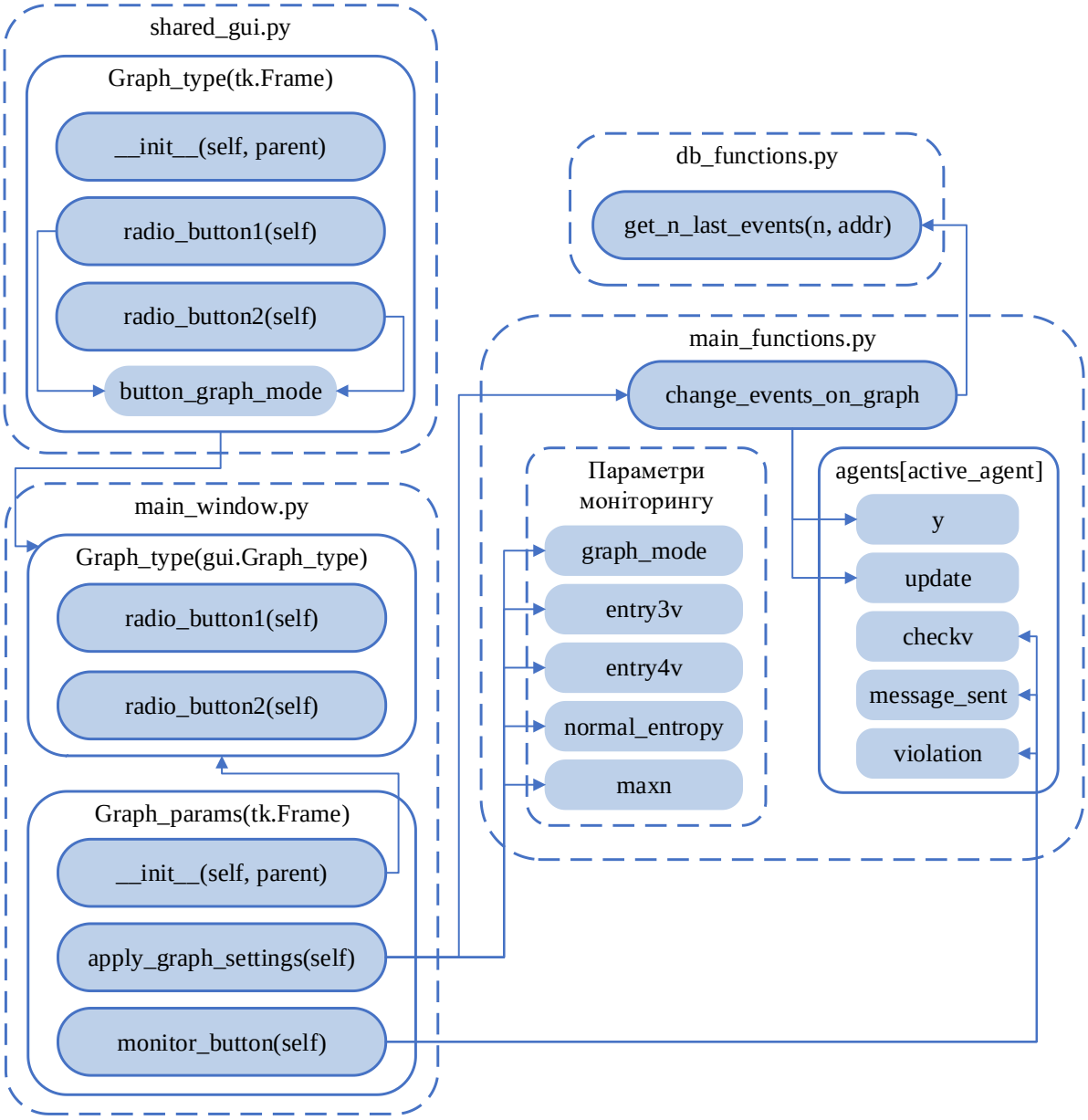


Рисунок 3.10 – Схема роботи панелі із параметрами графіка зміни ентропії на головному вікні

Клас «Tcp_client_thread» наслідує клас «threading.thread», що виконує вказані дії в окремому потоці. Цей клас містить функцію «run», яка виконується після запуску потоку і її можна перевизначити у класі, що наслідує «threading.thread». Клас «Tcp_client_thread» має такі функції:

- «__init__» – ініціалізація класу. У цій функції спочатку виконується ініціалізація класу, що наслідується. Далі створюється змінна «addr», у яку записується адреса агента, що передається в якості параметра під час створення об’єкта класу.

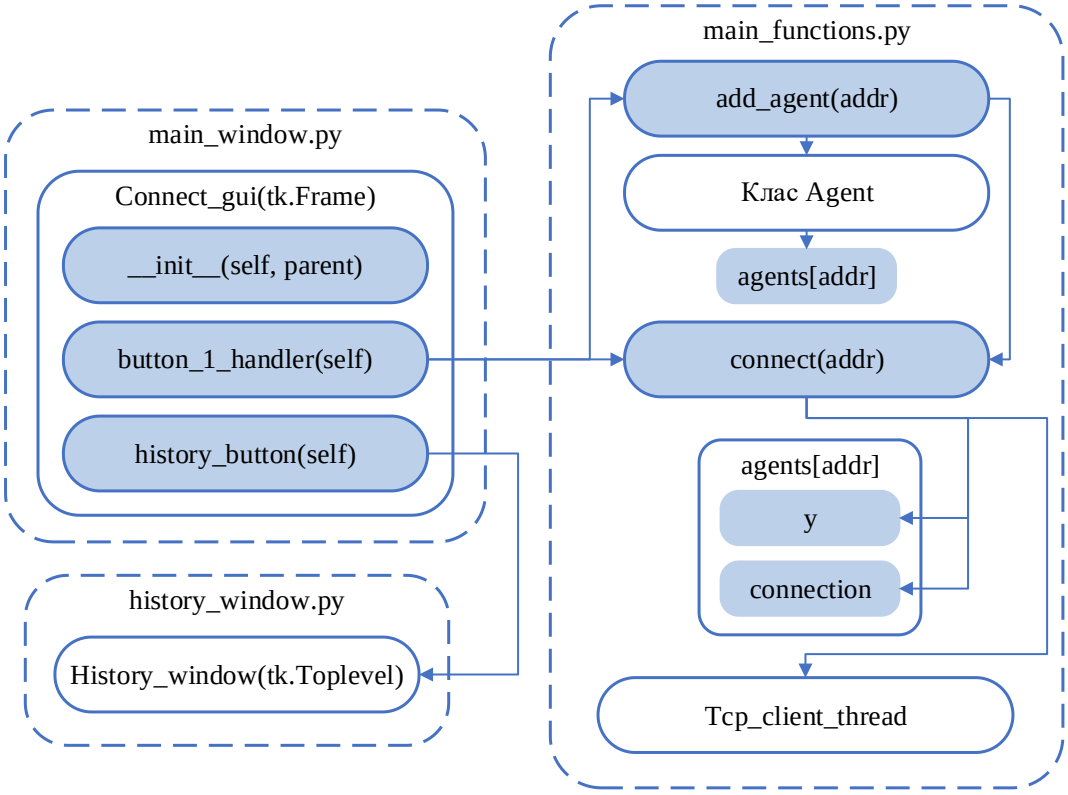


Рисунок 3.11 – Схема класу «Connect_gui»

- «run» – у цій функції описано роботу TCP клієнта. Виконується підключення до агента із використання модуля «socket» та присвоєння значення «True» змінній «connection» відповідного об’єкта класу «Agent». Використано конструкцію «try-except» для перехоплення виключення у разі помилки підключення та виведення відповідного повідомлення, а також встановлення значення «False» змінній «connection». Якщо встановлення з’єднання виконано успішно, тоді викликається функція «create_db_table» із файлу «db_functions.py», що

спочатку викликає функцію «check_if_table_exists», яка виконує SQL запит для перевірки наявності таблиці із назвою, що відповідає адресі агента. Якщо такої таблиці немає, тоді виконується SQL запит для створення таблиці в базі даних. Після виклику функції «create_db_table» виконується отримання даних від агента розміром 66 байтів. Такий розмір має одне повідомлення від агента, яке містить значення ентропії і об’єкт «datetime», який містить дату і час розрахунку цього значення ентропії. Блок коду, що призначений для отримання даних, записано в конструкції «try-except» для перехоплення виключень у разі розриву з’єднання. У такому випадку закривається сокет, встановлюється значення «False» для змінних «connection» і «checkv», а також виводиться повідомлення про закриття з’єднання. Після отримання даних від агента викликається функція «add_new_value».

- «add_new_value» – функція, яка додає отримане значення ентропії до списку «y». Якщо довжина цього списку рівна змінній «maxn», тоді перше значення у списку видаляється. Під час зміни списку «y» використовується конструкція «with agents[self.addr].varlock», де змінна «self.addr» – адреса агента, до якого виконано підключення. Після оновлення списку «y» встановлюється значення «True» для змінної «update». Також викликаються функція «update_db» із файлу «db_functions.py» і функція «check». Функція «update_db» виконує SQL запит для запису отриманих значень ентропії та дати і часу в базу даних.

- «check» – функція, що виконує перевірку перевищення встановлених допустимих меж ентропії. Якщо отримане значення перевищує ці межі, тоді це значеня записується в змінну «vi_val», для змінної «violation» встановлюється значення «True», а для змінної «checkv» – «False». Також адреса агента записується в чергу «q» для відправки повідомлення про перевищення.

- «close_connection» – функція, що закриває сокет, викликається після закриття програми.

Схема описаного класу показана на рисунку 3.12.

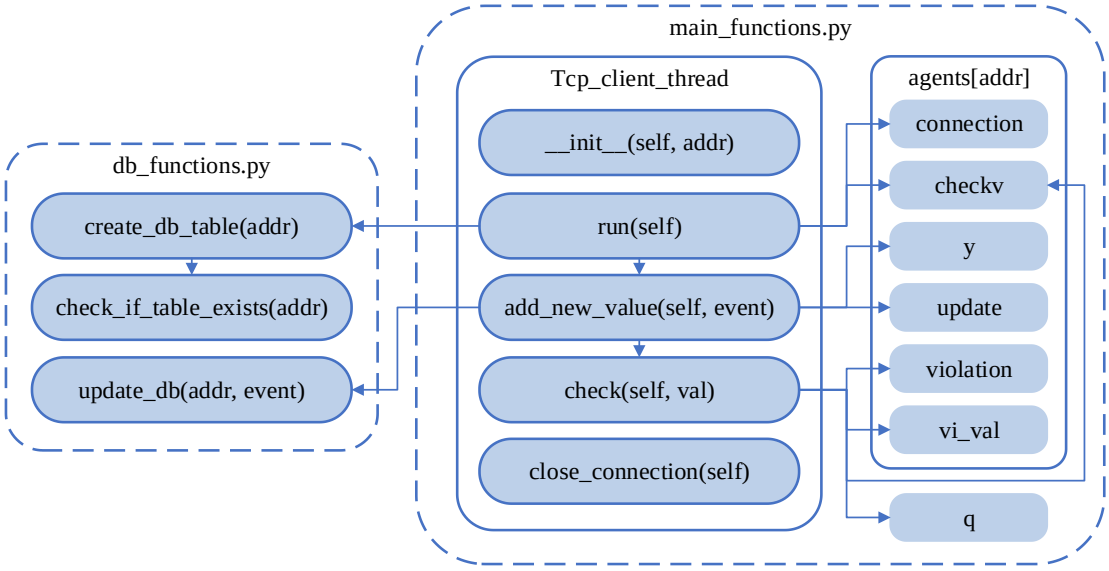


Рисунок 3.12 – Схема класу «Tcp_client_thread»

Панель, що призначена для введення параметрів та ввімкнення відправки повідомлень на пошту і Telegram, описана в класі «Message_gui». Цей клас містить такі функції:

- «__init__» – ініціалізація класу. У цій функції створюються кнопки перемикання між відправкою повідомлень поштою і в Telegram, кнопка увімкнення та вимкнення відправки, а також текстові поля і об’єкти типу «entry», що призначені для введення параметрів. Також створюється булева змінна «button_message_method» зі значенням «True», що позначає обрану вкладку панелі – відправка на електронну пошту, або в Telegram.
- «email_button» – функція, що викликається після натискання кнопки «Email». Ця функція змінює значення текстових панелей, а також розташування цих панелей і об’єктів «entry». Крім того, встановлюється значення «True» для змінної «button_message_method».
- «telegram_button» – функція, що викликається після натискання кнопки «Telegram». Ця функція змінює значення текстових панелей, а також розташування цих панелей і об’єктів «entry». Крім того, встановлюється значення «False» для змінної «button_message_method».
- «message_button» – функція, що викликається після натискання кнопки «Увімкнути/Вимкнути повідомлення». Код цієї функції записано в конструкції «try-except» для перехоплення виключень і виведення відповідних повідомлень, якщо

користувач увів неправильні дані. Якщо змінна «button_message_method» має значення «True» і змінна «mail_send» – «False» (панель у режимі параметрів відправки на електронну пошту, відправка на пошту вимкнена), тоді введені користувачем дані в об’єктах «entry» записуються в змінні «server», «sender_email», «password», «receiver_email», а також назва кнопки змінюється на «Вимкнути повідомлення» і для змінної «mail_send» встановлюється значення «True». Якщо змінна «mail_send» мала значення «True», тоді змінні server», «sender_email», «password», «receiver_email» очищуються і для змінної «mail_send» встановлюється значення «False», а назва кнопки змінюється на «Увімкнути повідомлення». Якщо змінні «button_message_method» і «telegram_send» мають значення «False» (тобто панель у режимі Telegram і відправка в Telegram вимкнена), тоді введені користувачем дані в об’єктах «entry» записуються в змінні «chat_id» і «token», назва кнопки змінюється на «Вимкнути повідомлення» і для змінної «telegram_send» встановлюється значення «True». Якщо змінна «telegram_send» мала значення «True», тоді змінні «chat_id» і «token» очищуються і для змінної «telegram_send» встановлюється значення «False», а назва кнопки змінюється на «Увімкнути повідомлення».

Схема описаного класу показана на рисунку 3.13.

Вікно історії значень ентропії описано в класі «History_window» у файлі «history_window.py». Цей клас наслідує клас «Tkinter Toplevel» та містить такі функції:

- У функції «__init__» корегуються параметри вікна, а також створюються об’єкти класів, що описують його окремі компоненти. Також створюється кнопка «Показати графік».
- «no_data» – функція, що очищує графік та показує на ньому повідомлення «Немає даних».
- «on_closing» – функція, що викликається під час закриття вікна. Встановлюється значення «False» для змінної «history_opened» батьківського об’єкту.

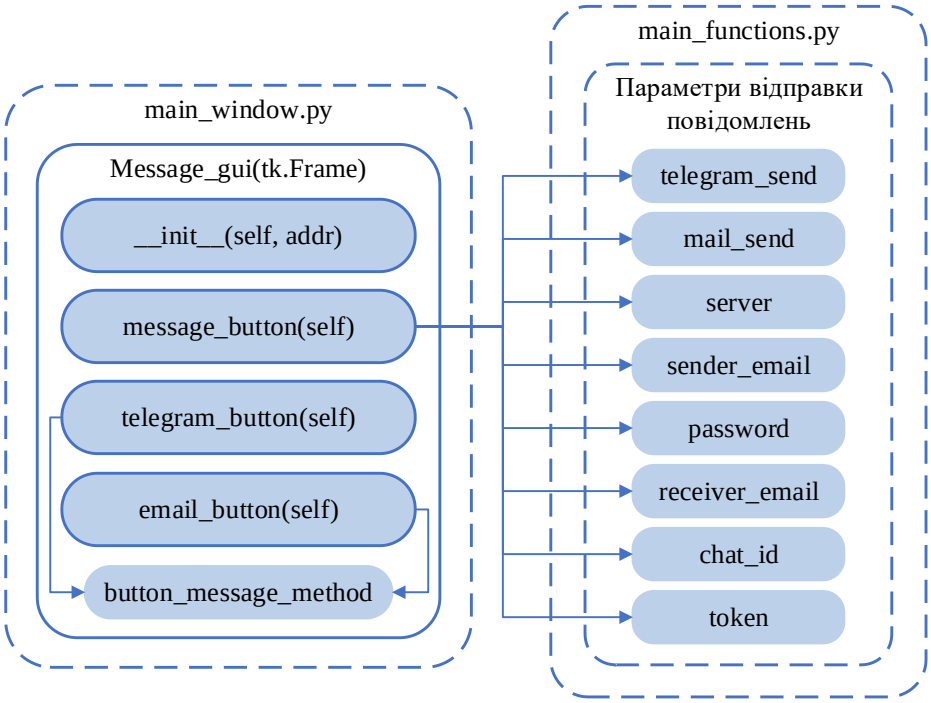


Рисунок 3.13 – Схема класу «Message_gui»

- «show» – функція, що викликається після натискання кнопки «Показати графік». Спочатку отримуються дата і час початку у кінця періоду, для якого будуть виведені значення ентропії. Для цього викликаються функції «get_date» об’єктів «date entry» і «get_time» об’єктів класу «DateTime_entry». Отримані дата і час поєднуються в об’єкт «datetime» використовуючи функцію «combine». У результаті отримано два об’єкти «datetime», які передаються у функцію «get_events_by_date» із файлу «db_functions.py» разом із введеною адресою агента, якщо функція «check_if_table_exists» повернула значення «True» для цієї адреси. Функція «get_events_by_date» виконує SQL запит для отримання списку подій в межах вказаного проміжку дати і часу. Отримані дані у циклі «for» записуються в список «y». Якщо обрано режим графіку «ентропія від кількості подій», тоді список «y» передається у функцію «show_graph_n» класу «Graph_gui». Якщо обрано режим графіку «ентропія від дати/часу», тоді додатково формується список «x», у який записуються дата і час для кожного значення ентропії та викликається функція «show_graph_date» класу «Graph_gui». Якщо кількість значень ентропії менша за 2, тоді викликається функція «no_data». Увесь код функції записано в конструкції «try-except» для перехоплення виключень, якщо користувач увів неправильні дані.

Схема описаного класу показана на рисунку 3.14.

Клас «Graph_gui» у файлі «history_window.py» наслідує одноіменний клас із файлу «shared_gui.py» та має такі функції:

- «__init__» – ініціалізація класу. У цій функції корегується розмір та положення графіка у вікні.
- «show_graph_n» – функція, що показує графік ентропії від кількості подій. Виконується коректування полів графіка, далі викликається функція «plot_graph», якій передається список «y», побудований графік відображається викликом функції «draw» об’єкта «FigureCanvasTkAgg», а також викликається функція «show_min_max_avg».
- «show_graph_date» – функція, що показує графік ентропії від дати і часу. Виконується коректування полів графіка, побудова графіка за введеними списками «x» та «y», викликаються функції «set_y_axis_params» та «set_x_axis_date_params»

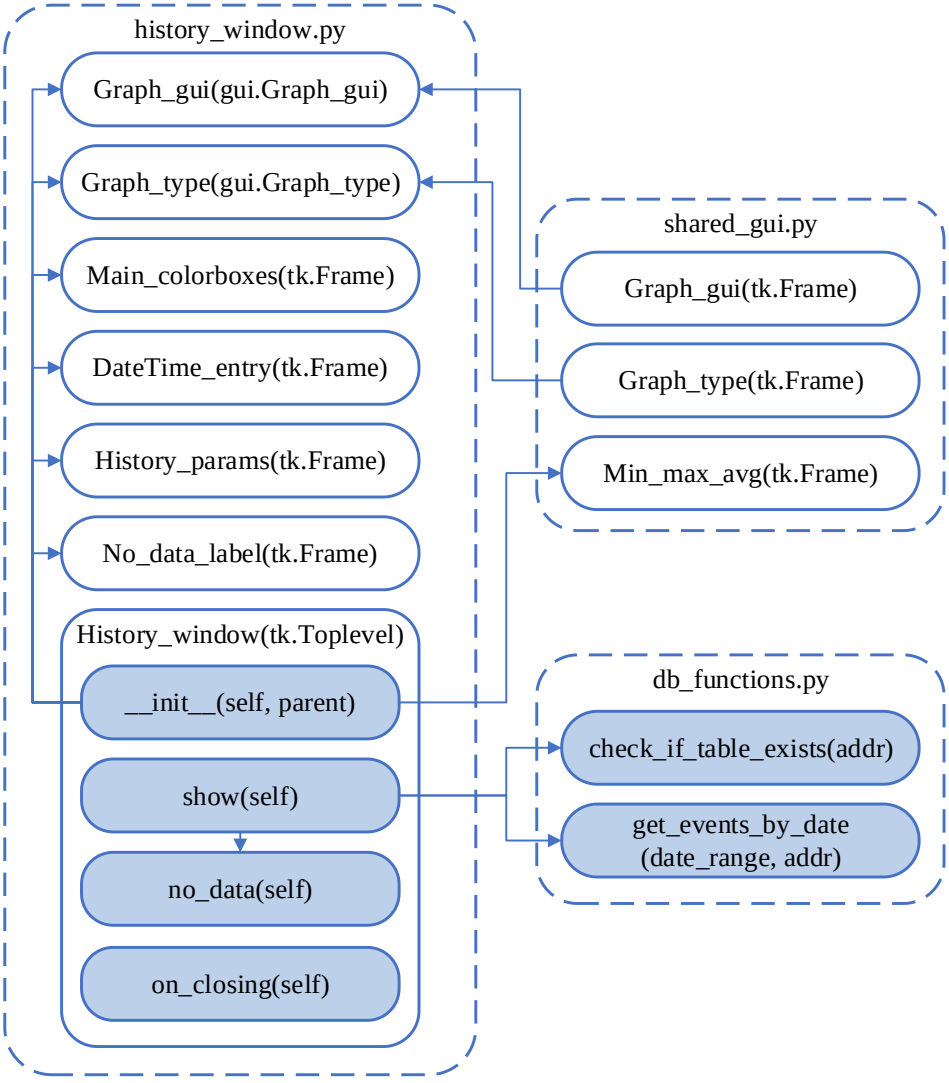


Рисунок 3.14 – Схема класу «History_window»

для встановлення параметрів осей. Побудований графік відображається викликом функції «draw» об’єкта «FigureCanvasTkAgg», а також викликається функція «show_min_max_avg».

- «set_x_axis_date_params» – функція, що встановлює параметри осі «X», на якій відображається дата і час. Викликається функція форматування дати «DateFormatter» із модуля «matplotlib.dates», а також функції «AutoMinorLocator», «MaxNLocator» із модуля «matplotlib.ticker», що призначені для встановлення кількості позначок на осі. Також встановлюється вирівнювання і розмір тексту позначок використовуючи функцію «setp» об’єкта «pyplot» із модуля «matplotlib». Крім того, встановлюються межі осі «X», які відповідають початку і кінцю вказаного користувачем проміжку часу використовуючи функцію «set_xlim».

Схема описаного класу показана на рисунку 3.15.

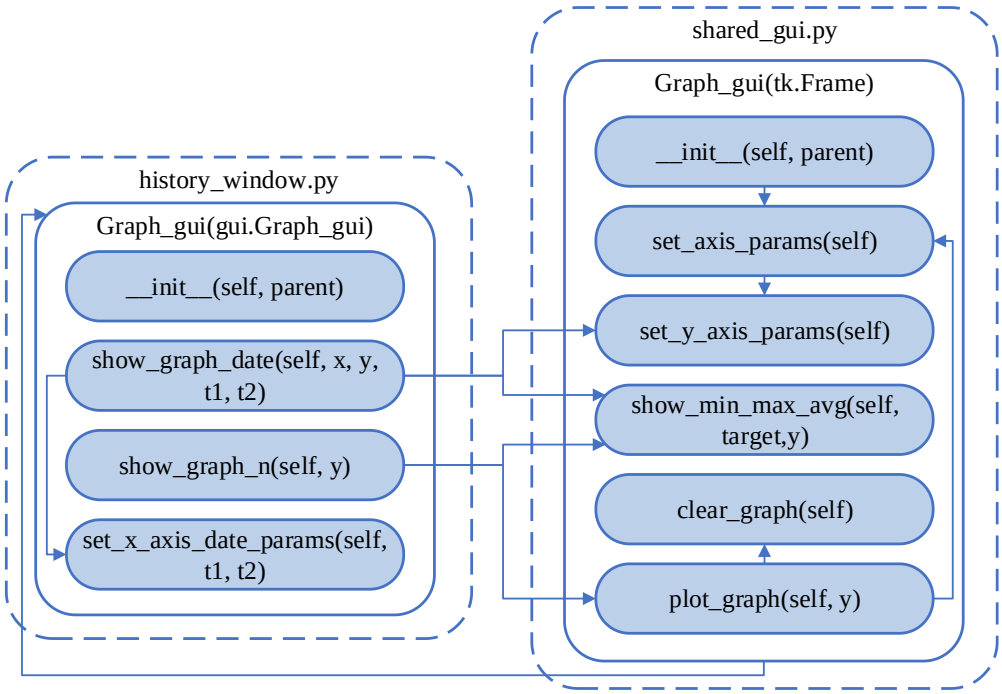


Рисунок 3.14 – Схема класу «Graph_gui»

Клас «Graph_type» наслідує однойменний клас із файлу «shared_gui.py». У функції «__init__» змінюється значення текстових полів та розташування елементів на панелі.

Клас «Main_colorboxes» описує панель із заголовком вікна. У функції «__init__» встановлюється розмір та колір панелі, а також створюється текстова панель (об’єкт класу «Label») із назвою вікна.

Клас «DateTime_entry» описує панель, що призначена для введення дати і часу. У функції «__init__» створюється об’єкт «DateEntry» із модуля «tkcalendar», який призначений для вводу дати, а також три об’єкти «Spinbox» із модуля «Tkinter ttk», що призначені для введення годин, хвилин і секунд. Також клас має функцію «get_time», яка формує об’єкт «time» модуля «datetime» із чисел, які користувач записав в об’єктах «Spinbox».

Клас «History_params» описує панель, що призначена для введення параметрів пошуку значень ентропії в базі даних для відображення їх на графіку. У функції «__init__» створюються текстові поля, а також об’єкт «Entry», що призначений для введення адреси агента, і два об’єкти класу «DateTime_entry», які призначені для введення початку і кінця проміжку дати і часу.

Клас «No_data_label» описує панель, що відображає на графіку повідомлення «Немає даних». У функції «__init__» встановлюється розмір та колір панелі, а також створюється текстова панель із відповідним текстом.

Висновки за розділом 3

У цьому розділі описано процес розробки програмного засобу. Він складається з двох частин – агента у хмарі, який отримує записи журналу операційної системи та розраховує значення ентропії, а також аналізатора, що показує значення ентропії на графіку і виконує перевірку, чи не перевищують ці значення встановлені користувачем межі. Якщо виявлено перевищення, тоді відповідне повідомлення надсилається на електронну пошту та Telegram.

Було описано всі функції та класи, з яких складаються зазначені компоненти програмного засобу. Код агента міститься в одному файлі, а код аналізатора розділено на функціональні частини та розміщено в сімох файлах, окремі компоненти програми описані в різних класах.

РОЗДІЛ 4
ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАСОБУ

4.1 Моніторинг ентропії та перегляд історії значень

Головне вікно програми-аналізатора показано на рисунку 4.1. У цьому вікні виконується підключення до агента, вводяться параметри моніторингу, відправки повідомлень про перевищення встановлених допустимих значень, а також відображається графік зміни ентропії.

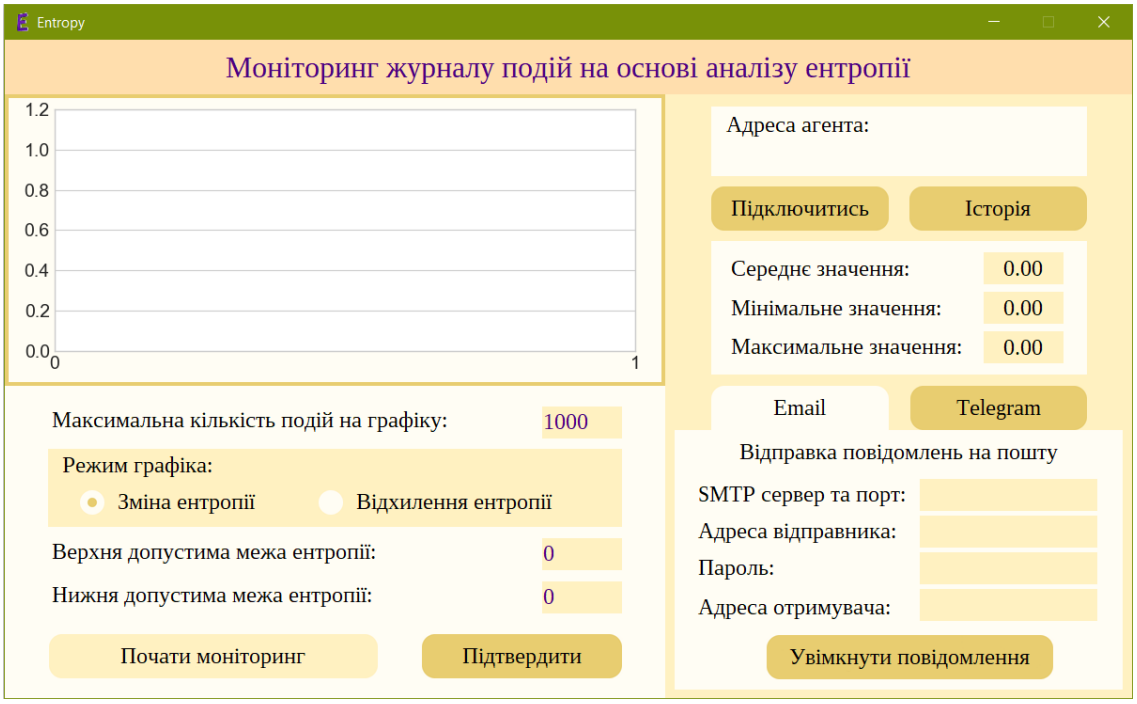


Рисунок 4.1 – Головне вікно аналізатора

Після виконання підключення на графіку відображаються значення ентропії, що отримуються від агента (рис. 4.2). Користувач має можливість змінити кількість подій на графіку, для яких відображається значення ентропії, а також режим графіку – «Зміна ентропії» або «Відхилення ентропії». Крім того, програма показує середнє, мінімальне та максимальне значеня ентропії на графіку.

Рисунок 4.2 – Графік зміни ентропії на головному вікні

Для активації моніторингу користувач має ввести верхню і нижню межу ентропії для режиму графіка «Зміна ентропії», підтвердити зміни, натиснувши кнопку «Підтвердити», та натиснути кнопку «Почати моніторинг». Після цього назва кнопки змінюється на «Перевищення не виявлено» (рис. 4.3).

Рисунок 4.3 – Виконання моніторингу

Для активації відправки повідомлень про перевищення встановлених значень ентропії користувач має ввести SMTP сервер та порт, адресу відправника та пароль для цієї адреси, а також адресу отримувача повідомлень (рис. 4.4).

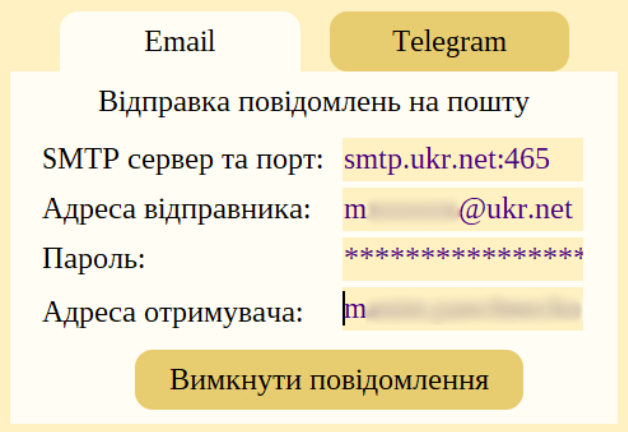


Рисунок 4.4 – Параметри відправки повідомлень на електронну пошту

Відправка повідомлень показана на прикладі сервісу «Ukr.net». Цей сервіс дозволяє створювати окремі паролі для доступу зовнішніх програм до електронної пошти. Створення пароля доступно в налаштуваннях безпеки у вкладці «Керування IMAP-доступом» (рис. 4.5).

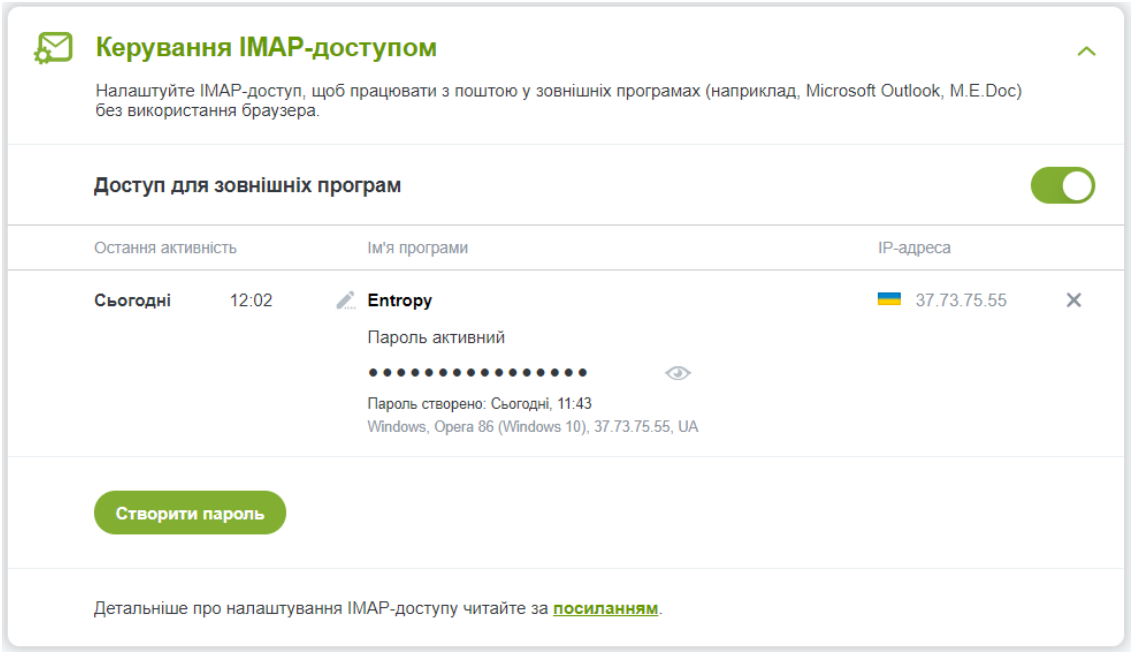


Рисунок 4.5 – Налаштування доступу зовнішніх програм до поштового сервісу «Ukr.net»

Також програма має можливість відправки повідомлень в Telegram із використанням бота. Для активації необхідно ввести API токен бота, а також chat ID користувача, якому будуть надсилатись повідомлення (рис. 4.6).

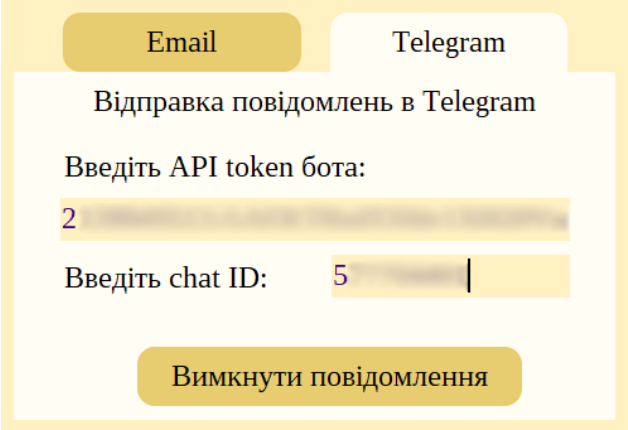


Рисунок 4.6 – Параметри відправки повідомлень в Telegram

Для створення нового бота в Telegram використовується бот «BotFather». Після введення назви нового бота йому надається API токен, що дозволяє отримувати доступ цього бота через HTTP API (рис. 4.7).

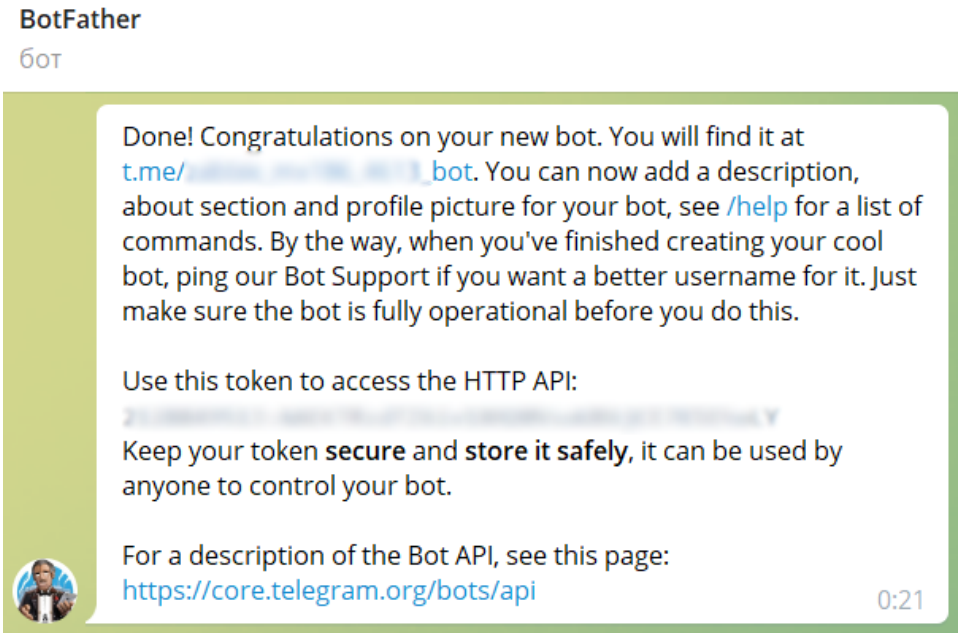


Рисунок 4.7 – Створення нового бота в Telegram

Якщо значення ентропії виходить за межі встановлених порогів безпеки, тоді з’являється повідомлення «Перевищення» та моніторинг зупиняється. Для відновлення моніторингу потрібно натиснути на кнопку «Перевищення» (рис. 4.8). Також повідомлення про перевищення разом із графіком зміни ентропії надсилається в Telegram (рис. 4.9) та на електронну пошту (рис. 4.10).

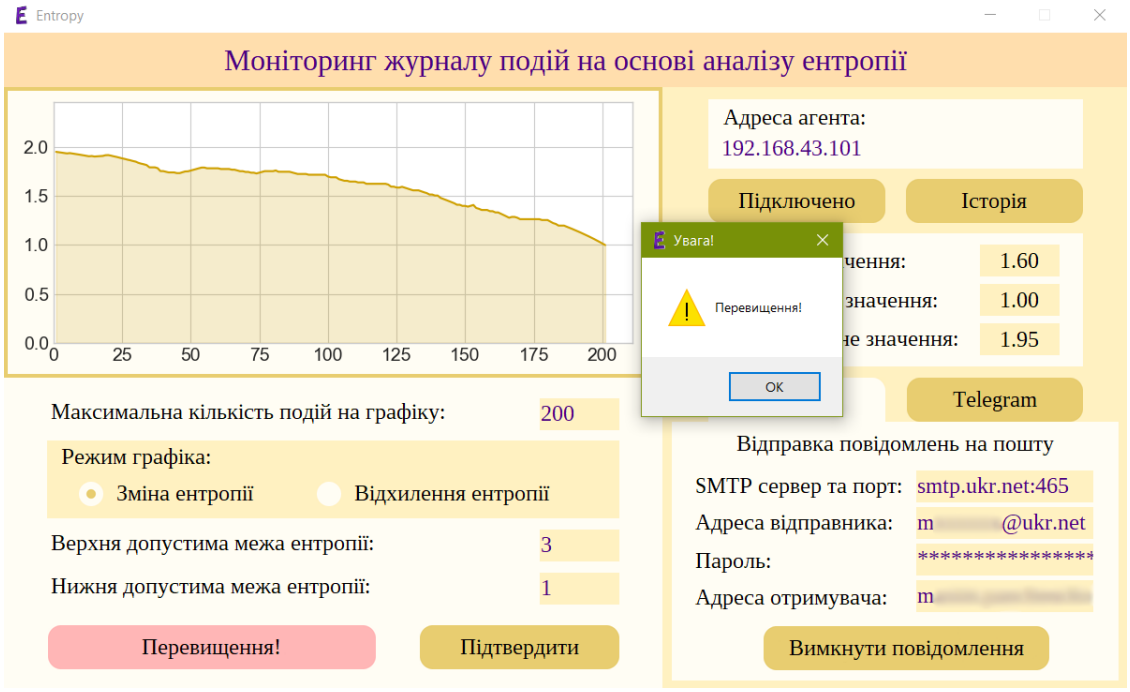


Рисунок 4.8 – Повідомлення про перевищення встановлених порогів безпеки у програмі

Рисунок 4.9 – Повідомлення про перевищення встановлених порогів безпеки в Telegram

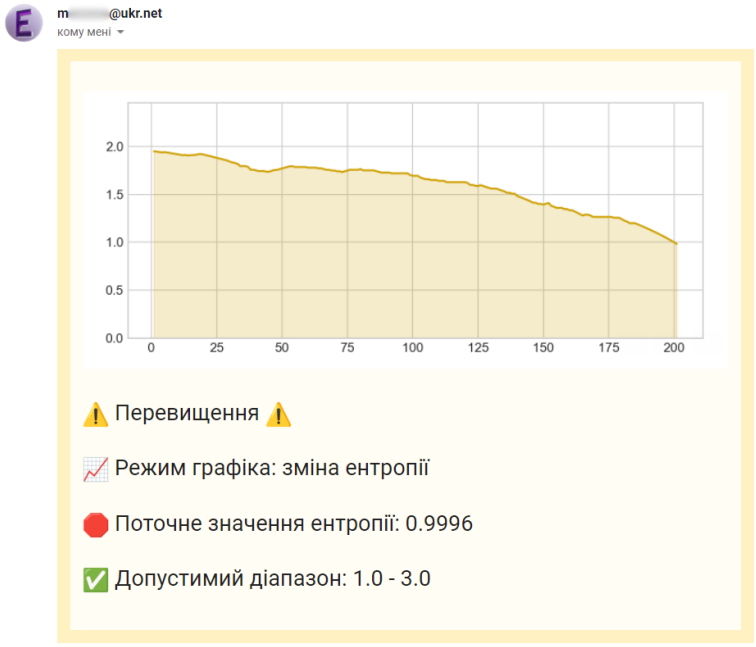


Рисунок 4.9 – Повідомлення про перевищення встановлених порогів безпеки на електронній пошті

Якщо користувач обрав режим графіку «Відхилення», тоді пропонується ввести значення ентропії за нормальної роботи системи, а також допустиме відхилення. На графіку буде відображатись відхилення від вказаного значення. На рисунку 4.10 показано графік у такому режимі роботи.

Рисунок 4.10 – Робота програми у режимі «Відхилення ентропії»

Якщо відхилення ентропії перевищує встановлене значення, тоді виводиться повідомлення про перевищення та призупиняється моніторинг (рис. 4.11), а також відправляється повідомлення на електронну пошту (рис. 4.12) та Telegram.

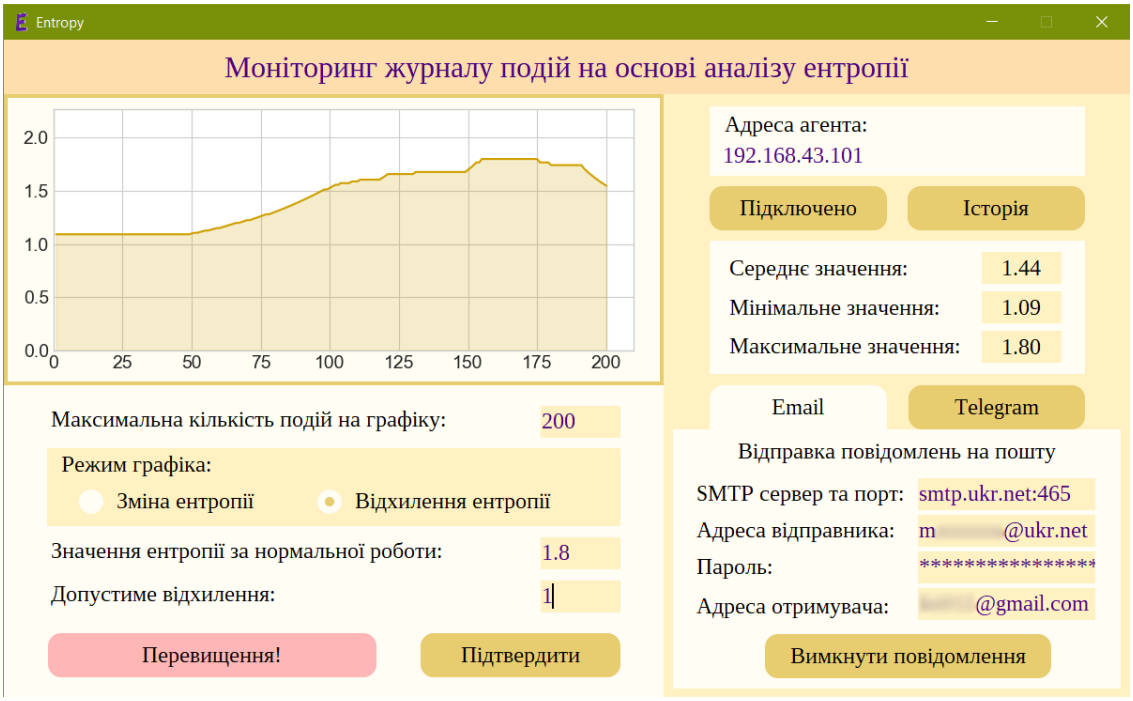


Рисунок 4.11 – Перевищення допустимого відхилення ентропії від нормального значення

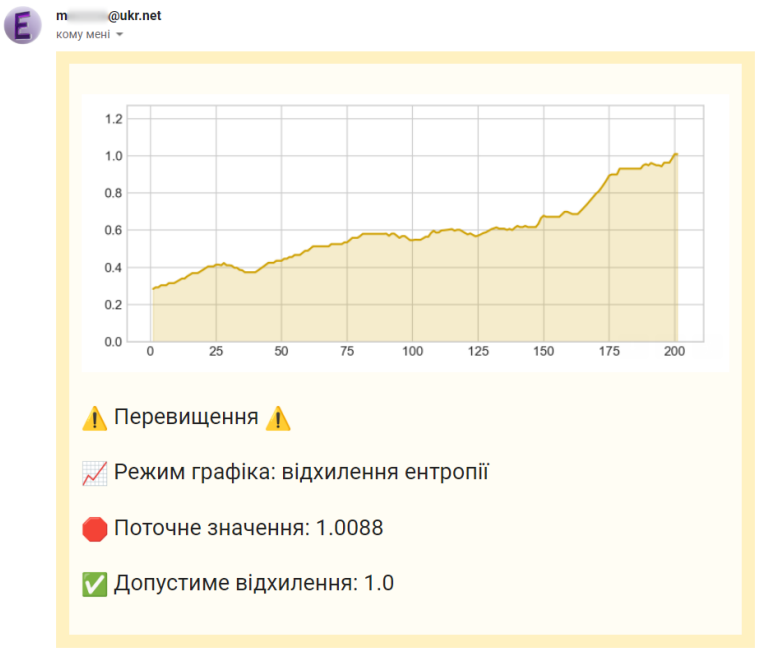


Рисунок 4.12 – Повідомлення про перевищення допустимого відхилення ентропії на електронній пошті

Програма також дозволяє переглядати попередні значення ентропії, що зберігаються в базі даних. Для цього потрібно натиснути кнопку «Історія» на головному вікні програми. Після цього відкриється вікно, у якому пропонується ввести адресу агента, для якого буде відображено графік ентропії, а також часовий діапазон. У даному вікні є можливість обрати два режими графіка: «Ентропія від кількості подій» (рис. 4.13) та «Ентропія від дати/часу» (рис. 4.14). Також у цьому вікні показано середнє, мінімальне та максимальне значення ентропії на графіку.

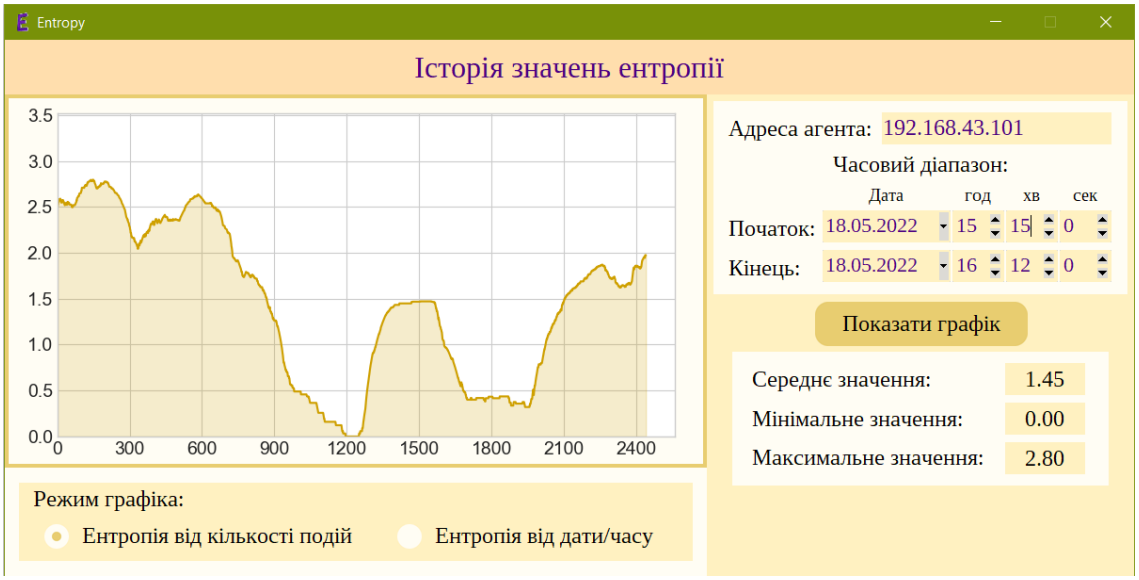


Рисунок 4.13 – Графік історії ентропії від кількості подій

Рисунок 4.14 – Графік історії ентропії від дати і часу

4.2 Виявлення аномалій в журналі подій

Перший експеримент – аналіз журналу подій із записами служби Sysmon. На рисунку 4.15 показано графік зміни ентропії під час звичайної роботи системи з увімкненою службою Sysmon. Середнє значення ентропії – 2,25, мінімальне – 1,6, максимальне – 2,7.

Рисунок 4.15 – Графік зміни ентропії під час нормальної роботи системи

На наступному етапі досліджень виконали моделювання несанкціонованих дії в системі з активною службою Sysmon. В рамках реалізації даного етапу було виконано запис великої кількості файлів та створено велику кількість процесів.

На рисунку 4.16 показано графік зміни ентропії під час запису великої кількості файлів. Такий же графік отримано у випадку створення великої кількості процесів. Під час запису файлів ентропія дорівнює нулю, бо у вікні є лише повідомлення про цю подію і її ймовірність дорівнює 1.

Таким чином, отримані результати дозволили зробити висновок, що аналіз значень ентропії системи, які розраховані на основі записів із журналу служби Sysmon дозволяє одразу виявити аномальні події в системі, зокрема запис великої кількості файлів, створення великої кількості процесів в системі.

Крім того, велику кількість однакових записів в журналі можуть генерувати і інші події в системі, що не відносяться до несанкціонованих, наприклад, оновлення чи встановлення програм.

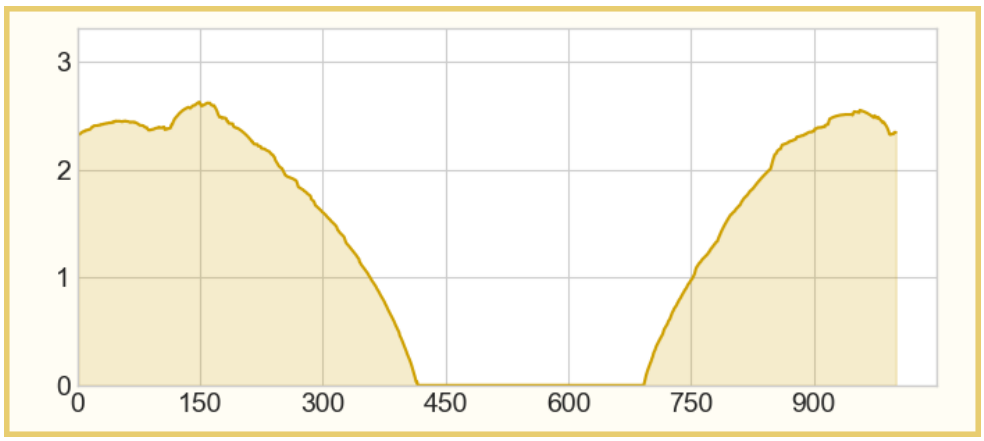


Рисунок 4.16 – Графік зміни ентропії під час запису великої кількості файлів

Також було виконано ще один тип атаки, який навпаки спричиняє протоколювання великої кількості різних подій в журналі. Результат атаки включає в себе віддалене підключення до цільової системи та виконання ескалації привілеїв. На рисунку 4.17 показано графік зміни ентропії під час нормальної роботи системи та під час виконання атаки. Середнє значення – 1,98, мінімальне – 1,53, максимальне – 2,72.

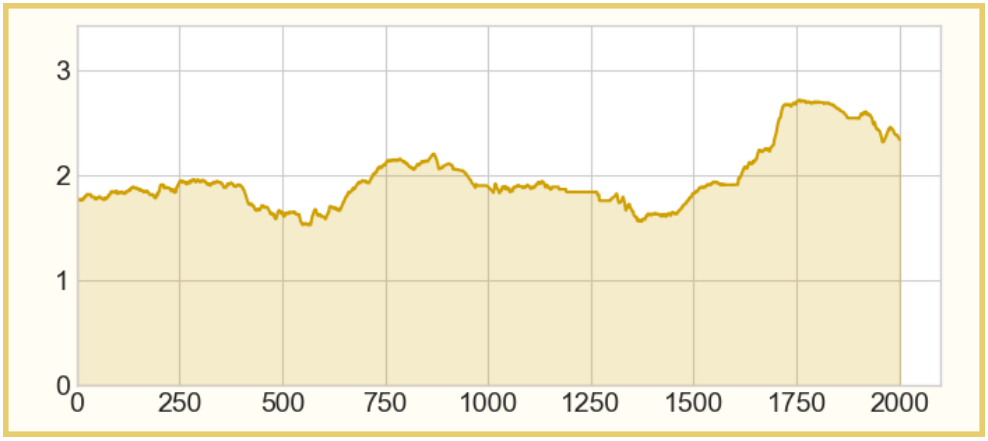


Рисунок 4.17 – Графік зміни ентропії під час виконання атаки

Під час виконання атаки значення ентропії було максимальним за весь період спостережень, що зумовлено протоколюванням подій безпеки, зокрема таких, як створення файлів, модифікація реєстру, виконання команд в консолі та створення процесів.

Порівняємо наведений графік із середнім значенням ентропії на проміжку, що відповідає нормальній роботі системи. Приблизне середнє значення на цьому проміжку – 1,9. Тоді графік відхилення ентропії від вказаного значення буде виглядати так, як показано на рисунку 4.18. Під час нормальної роботи системи відхилення ентропії не перевищує значення 0,4, а під час виконання атаки становить 0,81.

Рисунок 4.18 – Відхилення ентропії від еталонних значень

На основі вище означеного було зроблено висновок, що описані атаки суттєво вливають на значення ентропії журналу подій, відповідно велике відхилення від еталонних значень ентропії може вказувати на виконання несанкціонованих дій в системі.

4.3 Можливості для вдосконалення

Під час виконання тестувань було виявлено, що деякі інциденти безпеки можуть генерувати декілька однакових повідомлень в журналі подій. У такому випадку значення ентропії не буде близьке до нуля, тому ці події можуть не бути виявлені розробленим програмним засобом.

Наприклад, під час завантаження файлів з інтернету браузер створює додатковий файловий потік (file stream), що є однією із можливостей файлової системи NTFS. Цей потік має назву «Zone.Identifier» та призначений для позначення

файлів, завантажених з інтернету, але цей потік може також використовуватись зловмисниками для приховування шкідливих файлів [18]. Зазвичай цей потік вміщує в себе код зони (zoneid), яка позначає рівень довіри до файлу, а також адресу, за якою завантажено файл (рис. 4.19).

```
Windows PowerShell
PS C:\Users\maxim> Get-Content -path C:\Users\maxim\Downloads\125961387.pdf
-stream Zone.Identifier
[ZoneTransfer]
ZoneId=3
ReferrerUrl=https://scholar.google.com/
HostUrl=https://www.atlantis-press.com/article/125961387.pdf
```

Рисунок 4.19 – Вміст файлового потоку «Zone.Identifier»

Служба Sysmon буде реєструвати створення таких файлів як дві події – «File created» та «File stream created». Тому під час створення великої кількості файлів, що мають додатковий файловий потік, журнал подій служби Sysmon буде заповнений цими двома подіями (рис. 4.20).

Operational Событий: 50 647 (!) Есть новые события

Уровень	Дата и время	Источник	Код события	Категория задачи
Сведения	23.05.2022 20:15:55	Sysmon	15	File stream created (rule: FileCr...
Сведения	23.05.2022 20:15:55	Sysmon	11	File created (rule: FileCreate)
Сведения	23.05.2022 20:15:55	Sysmon	15	File stream created (rule: FileCr...
Сведения	23.05.2022 20:15:55	Sysmon	11	File created (rule: FileCreate)
Сведения	23.05.2022 20:15:55	Sysmon	15	File stream created (rule: FileCr...
Сведения	23.05.2022 20:15:55	Sysmon	11	File created (rule: FileCreate)
Сведения	23.05.2022 20:15:55	Sysmon	15	File stream created (rule: FileCr...
Сведения	23.05.2022 20:15:55	Sysmon	11	File created (rule: FileCreate)
Сведения	23.05.2022 20:15:55	Sysmon	15	File stream created (rule: FileCr...
Сведения	23.05.2022 20:15:55	Sysmon	11	File created (rule: FileCreate)
Сведения	23.05.2022 20:15:55	Sysmon	15	File stream created (rule: FileCr...
Сведения	23.05.2022 20:15:55	Sysmon	11	File created (rule: FileCreate)
Сведения	23.05.2022 20:15:55	Sysmon	15	File stream created (rule: FileCr...
Сведения	23.05.2022 20:15:55	Sysmon	11	File created (rule: FileCreate)

Рисунок 4.20 – Повідомлення у журналі служби Sysmon про створення файлів, що мають додатковий файловий потік

У цьому випадку дві події будуть рівноймовірними, тоді значення ентропії буде максимальним для множини повідомлень із двох елементів та рівним 1 (рис. 4.21).

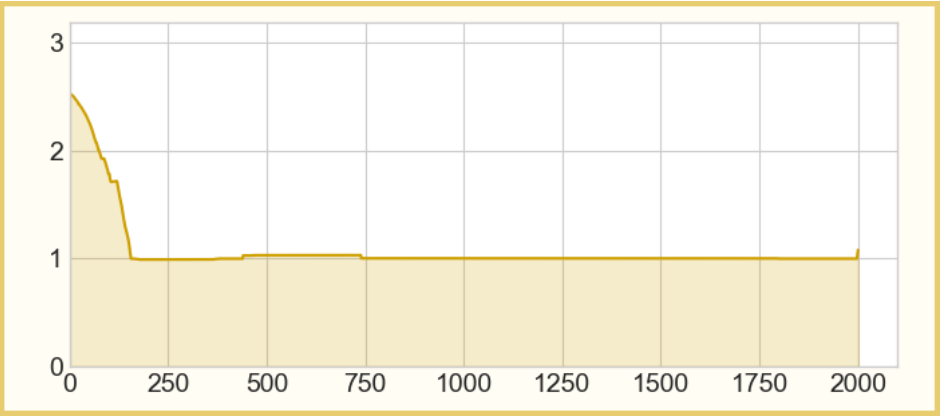


Рисунок 4.21 – Графік зміни ентропії під час запису великої кількості файлів, що мають додатковий файловий потік

Для виявлення великої кількості повторень декількох записів в журналі подій можна додати до розробленого програмного засобу розрахунок дисперсії вибірки значень ентропії та її контроль на перевищення встановлених користувачем меж. Дисперсія показує ступінь розсіювання випадкової величини відносно її середнього значення [19, с. 41].

Також ще одним варіантом вдосконалення є можливість підключатись до декількох агентів та виконувати моніторинг значень ентропії, що надходять із декількох хостів. Використовуючи розроблений програмний засіб це можна зробити лише якщо відкрити декілька екземплярів програми, що створює незручності.

Під час розробки програми інформацію, що пов’язана з агентами, а також код TCP сервера було виокремлено в окремі класи, що дозволяє створювати необмежену кількість об’єктів цих класів. Тому таке вдосконалення можна зробити із мінімальними змінами коду програми – необхідно лише додати графічний інтерфейс, що дозволяє підключатись до декількох агентів, а також написати функції, які будуть контролювати відображення інформації з різних агентів на головному вікні.

Крім того, можна розробити агент для журналів подій інших операційних систем. Для цього необхідно лише додати код, що призначений для отримання записів із журналів. Водночас розрахунок ентропії та відправка результатів обчислень залишаться незмінними.

Висновки за розділом 4

У цьому розділі описано тестування розробленого програмного засобу. Показано роботу усіх компонентів програми, а також виконано деякі типи атак на інформаційну систему. Отримані результати дозволили зробити висновки, що розроблена програма дозволяє виявляти описані атаки.

Також описано можливості вдосконалення розробленого програмного засобу. Деякі атаки можуть генерувати велику кількість декількох однакових подій, які можна виявити шляхом розрахунку дисперсії ентропії журналу подій, яка показує міру розсіювання значень випадкової величини відносно середнього. Також для вдосконалення програми можна розробити агент для інших операційних систем і додати можливість моніторингу декількох систем одночасно.

ВИСНОВКИ

У процесі виконання дипломного проектування розроблено програмний засіб, що дозволяє виконувати моніторинг журналу подій операційних систем сімейства Windows шляхом обчислення інформаційної ентропії.

Виконано аналіз нормативно-правової бази щодо безпеки хмарних обчислень та зроблено висновок, що моніторинг є одним із необхідних складових забезпечення інформаційної безпеки хмарних середовищ. Серед компонентів інформаційної системи, за якими необхідно виконувати моніторинг, зокрема, є журнали подій. Також розглянуто деякі програми моніторингу журналів подій та визначено, що деякі аномалії в журналі подій можуть залишитись невиявленими. На основі цього сформовано завдання дипломного проектування.

Досліджено процес розрахунку ентропії методом рухомого вікна для джерел, які постійно генерують нові повідомлення, а також розроблено програмну реалізацію цього методу на мові програмування Python.

Після цього обрано операційну систему Windows та проаналізовано складові журналу подій цієї операційної системи. Програми моніторингу для хмарних середовищ зазвичай складаються із двох компонентів – агента та аналізатора. Тому розроблено агент, який отримує записи журналу подій Windows, а також розраховує ентропію і надсилає аналізатору поточні дату та час разом із обчисленими значеннями з використанням TCP з’єднання.

Далі розроблено аналізатор, який отримує значення ентропії від агента, показує їх на графіку, а також зберігає їх разом із датою і часом в базі даних. Також у межах моніторингу аналізатор виконує перевірку перевищення встановлених користувачем допустимих значень ентропії. Якщо виявлено перевищення, тоді моніторинг зупиняється і повідомлення надсилається на електронну пошту та Telegram. Крім того, аналізатор дозволяє переглядати на графіку збережені в базі даних значення ентропії.

Після розробки програмного засобу виконано його тестування. Показано роботу кожного компонента програми, а також виконано деякі типи атак на інформаційну систему. У межах моделювання атак створено велику кількість файлів та процесів. Такі дії створюють велику кількість однакових записів у журналі служби Sysmon та можуть вказувати на локальні DoS атаки. Ті самі результати будуть у випадку створення великої кількості записів в реєстрі, що також може вказувати на несанкціоновані дії. Крім того, виконано атаку, яка передбачає встановлення несанкціонованого віддаленого з'єднання та підвищення привілеїв. Ця атака навпаки спричиняє протоколювання великої кількості різних подій в журналі. Виконані дослідження дозволили зробити висновок, що такі атаки можуть бути виявлені розробленим програмним засобом.

У кінці запропоновано можливі шляхи вдосконалення розробленої програми, що включають в себе можливість виявлення додаткових типів атак, виконання моніторингу одразу декількох систем та розробку агентів для інших операційних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про хмарні послуги : Закон України від 17.02.2022 р. № 2075–IX. URL: https://zakon.rada.gov.ua/go/2075-20 (дата звернення: 29.05.2022).

2. Про затвердження вимог у сфері електронних довірчих послуг та Порядку перевірки дотримання вимог законодавства у сфері електронних довірчих послуг : Постанова Каб. Міністрів України від 01.01.2020 р. № 992-2018-п. URL: https://zakon.rada.gov.ua/go/992-2018-%D0%BF (дата звернення: 29.05.2022).

3. EUCS – Cloud Services Scheme. ENISA, 2020. 245 p. (Draft. ENISA). URL: https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme (date of access: 10.06.2022).

4. Jansen W., Grance T. Guidelines on security and privacy in public cloud computing. Gaithersburg, MD : National Institute of Standards and Technology, 2011. URL: https://doi.org/10.6028/nist.sp.800-144 (date of access: 10.06.2022).

5. ISO/IEC 17788:2014. Information technology – Cloud computing – Overview and vocabulary. Effective from 2014-10-10. Official edition. 2014. 10 p. URL: https://www.iso.org/standard/60544.html (date of access: 10.06.2022).

6. Качинський А. Б., Стремецька М. С. Операційна аналітика як інструмент моніторингу даних та управління подіями систем забезпечення кібербезпеки. *Доповіді Національної академії наук України*. 2021. Вип. 1. С. 9–16. URL: https://doi.org/10.15407/dopovidi2021.01.009. (дата звернення: 03.06.2022).

7. Бєрковський В. В., Безсонов О. С. Аналіз та класифікація методів виявлення вторгнень в інформаційну систему. *Системи управління, навігації та зв'язку*. 2017. Вип. 3. С. 57–62. URL: http://nbuv.gov.ua/UJRN/suntz_2017_3_17 (дата звернення: 03.06.2022).

8. Eventlog Key – Win32 apps. *Microsoft Docs*. URL: https://docs.microsoft.com/en-us/windows/win32/eventlog/eventlog-key (date of access: 01.05.2022).

9. Makanju A., Zincir-Heywood A. N., Milios E. E. Investigating event log analysis with minimum apriori information. *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, Ghent, 27-31 May 2013. Ghent, 2013. P. 962–968. URL: https://dl.ifip.org/db/conf/im/im2013/MakanjuZM13.pdf (date of access: 03.06.2022).

10. Russinovich M., Garnier T. Sysmon - Windows Sysinternals. *Microsoft Docs*. URL: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon (date of access: 03.06.2022).

11. SwiftOnSecurity. Sysmon configuration file template with default high-quality event tracing. GitHub, 2022. URL: https://github.com/SwiftOnSecurity/sysmon-config (date of access: 03.06.2022).

12. Effective penetration testing with Metasploit framework and methodologies / F. Holik et al. *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*, Budapest, 19–21 November 2014. Budapest, 2014. URL: https://doi.org/10.1109/cinti.2014.7028682 (date of access: 10.06.2022).

13. Johnson A., Haddad R. J. Evading Signature-Based Antivirus Software Using Custom Reverse Shell Exploit. *SoutheastCon 2021*, Atlanta, GA, USA, 10–13 March 2021. Atlanta, 2021. URL: https://doi.org/10.1109/southeastcon45413.2021.9401881 (date of access: 10.06.2022).

14. Жураковський Ю. П., Полторак В. П. Теорія інформації та кодування: Підруч. для студ. вищ. техн. навч. закл. Київ : «Вища шк.», 2001. 255 с.

15. Gudkov O. Calculation algorithm for network flow parameters entropy in anomaly detection. *IT security for the next generation, international round*, Delft, 11–13 May 2012. Delft, 2012. URL: https://silo.tips/download/calculation-algorithm-for-network-flow-parameters-entropy-in-anomaly-detection. (date of access: 25.05.2022).

16. Галіцин В. К., Галиціна О. В., Камінський О. Є. Системний аналіз організації моніторингу хмарних платформ. *Моделювання та інформаційні системи в економіці : зб. наук. пр.* 2019. Вип. 98. С. 42–51. URL: https://ir.kneu.edu.ua:443/handle/2010/33939 (дата звернення: 31.03.2022).

17. Галіцин В. К., Камінський О. Є. Моніторинг хмарних сервісів, розгорнутих у багатохмарному середовищі. *Моделювання та інформаційні системи в економіці : зб. наук. пр.* 2017. Вип. 94. С. 160–169. URL: https://ir.kneu.edu.ua:443/handle/2010/24318 (дата звернення: 31.03.2022).

18. Kao D.-Y., Chen Y.-P., Chang E.-C. Event Observation of Date-time Stamps for ADS Reconstruction. *2019 21st International Conference on Advanced Communication Technology (ICACT)*, PyeongChang Kwangwoon_Do, Korea (South), 17–20 February 2019. 2019. URL: https://doi.org/10.23919/icact.2019.8701988 (date of access: 10.06.2022).

19. Найко Д. А., Шевчук О. Ф. Теорія ймовірностей та математична статистика. Вінниця : ВНАУ, 2020. 382 с. URL: http://socrates.vsau.org/repository/getfile.php/24513.pdf (дата звернення: 25.05.2022).

ДОДАТКИ

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИПЛОМНОЇ РОБОТИ

Статті у наукових фахових виданнях України

1. Панченко М., Бігдан А., Бабенко Т. та ін. Виявлення аномалій інформаційної безпеки на основі аналізу ентропії інформаційної системи. *Енергетика і автоматика*. 2022. Вип. 1. С. 72–81. URL: http://journals.nubip.edu.ua/index.php/Energiya/article/view/16054 (дата звернення: 08.06.2022).

Тези наукових доповідей

1. Панченко М., Даков С. Виявлення атаки password spraying шляхом аналізу журналу авторизації на базі ентропії. *Проблеми експлуатації та захисту інформаційно-комунікаційних систем* : Тези науково-практ. конф., м. Київ, 7–9 черв. 2022 р. Київ, 2022. С. 84–85.

ДОДАТОК Б
ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Агент – файл «WindowsEventLogs.py»

```
import win32evtlog, re, math, threading, queue, pickle, socket
from datetime import datetime

ip = '0.0.0.0'
sources=('System', 'Security','Application','Microsoft-Windows-
Sysmon/Operational','Windows PowerShell','Microsoft-Windows-PowerShell/Operational')

lock = threading.Lock()
q = queue.Queue()
itemsn={}
seq=[]
itemsf={}
w=300
dw=1
H=float(0)

def new_logs_event_handler(reason, context, evt):
    log = win32evtlog.EvtRender(evt, win32evtlog.EvtRenderEventXml)
    provider=re.search(r'<Provider Name=([^>]+)/>',log)
    if provider[1].find('Microsoft-Windows-Sysmon')>(-1):
        provider=re.search(r'([0-9]+)</EventID>',log)

    calc(provider[1])
    return 0

def calcH(f):
    if f!=0:
        h=float((-1)*f*math.log2(f))
    else: h=float(0)
    return h

def calc(provider):
    global H, w, dw, itemsn, itemsf, seq, connection_present
    lock.acquire()
    seq.append(provider)
    n=len(seq)
    itemsn.clear()

    if n>=w:
        for i, val in enumerate(seq):
            if i>(n-w-1) or n==w:
                if val in itemsn:
                    itemsn[val]+=1
                else:
                    itemsn[val]=1
            else:
                itemsn[val]=0

    if n==w:
        for i in itemsn:
            itemsf[i]=float(itemsn[i]/w)
```

```
            H+=calcH(itemsf[i])
    elif n>w:
        if not provider in itemsf:
            itemsf[provider]=float(0)
        if (n-w)==dw:
            H+=calcdH()
            for i in range(0,dw):
                seq.pop(0)
            now = datetime.now()
            data = [now, round(H,5)]
            print(data)
            q.put(data)
    lock.release()

def before(chseq):
    b=float(0)
    for i in chseq:
        b+=calcH(itemsf[i])
    return b

def now(chseq):
    now=float(0)
    for i in chseq:
        now+=calcH(float(itemsn[i]/w))
    return now

def calcdH():
    global itemsf
    chseq=[]
    l=len(seq)
    for i in range(0,dw):
        if not seq[i] in chseq: chseq.append(seq[i])
        if not seq[l-1-i] in chseq: chseq.append(seq[l-1-i])

    dh=(-1)*before(chseq)+now(chseq)

    for i in chseq:
        itemsf[i]=float(itemsn[i]/w)
        if itemsf[i]==0:
            itemsf.pop(i)
    return dh

def server():
    global connection_present
    s  = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    address = (ip, 1234)
    s.bind(address)
    s.listen(0)
    print('server started')
    while True:
        try:
            client, client_addr = s.accept()
            print('connected')
            while True:
                event = q.get()
                data=pickle.dumps(event)
                print(len(data))
                client.send(data)
                q.task_done()
        except:
            client.close()
            print('disconnected')

def main():
```

```
    subscriptions=[]
    for source in sources:
        subscriptions.append(win32evtlog.EvtSubscribe(source,
win32evtlog.EvtSubscribeToFutureEvents, None, Callback=new_logs_event_handler,
Context=None, Query=None))
    server()

if __name__=='__main__': main()
```

Аналізатор – файл «Entropy_client.py»

```
import tkinter as tk
import ctypes
import os
import main_window as window

def on_closing(root, w):
    w.close_connections()
    root.destroy()

def main():
    root = tk.Tk(className = 'Entropy')
    root.title('Entropy')
    if os.name == 'nt':
        ctypes.windll.shcore.SetProcessDpiAwareness(1)
        root.iconbitmap(default='Icon.ico')
    else:
        img = tk.PhotoImage(file='Icon.png')
        root.tk.call('wm', 'iconphoto', root._w, img)
    w = window.main_gui(root)
    w.pack(side="top", fill="both", expand=True)
    root.protocol("WM_DELETE_WINDOW", lambda: on_closing(root, w))
    root.mainloop()

if __name__=='__main__':
    main()
```

Аналізатор – файл «main_window.py»

```
import tkinter as tk
import matplotlib.animation as animation
import main_functions as f
import shared_gui as gui
import history_window as history
import relative_to_assets as assets

class Main_colorboxes(tk.Frame):
    def __init__(self,parent):
        tk.Frame.__init__(self, parent, bg = "#FFDEAD", height = 58, width = 1200)
        self.parent = parent
        self.parent.frame = tk.Frame(master = self.parent, bg="#FFF1C1", width = 497,
height = 642)
        self.parent.frame.place(x = 703, y = 58)
        self.label1 = tk.Label(master = self.parent, text = 'Моніторинг журналу подій
на основі аналізу ентропії',font = ("Tinos", 24), bg = '#FFDEAD',fg="#4B0082")
        self.label1.place(x = 0, y = 12, width = 1200, height = 36)
```

```
class Graph_gui(gui.Graph_gui):
    def __init__(self, parent):
        super().__init__(parent)
        self.ani = animation.FuncAnimation(self.fig, self.animate, interval=500)

    def animate(self, i):
        if f.active_agent != '':
            agent = f.agents[f.active_agent]
            with agent.varlock:
                y = agent.y
                if agent.update and len(y) > 1:
                    self.plot_graph(y)
                    self.show_min_max_avg(self.parent,y)
                    agent.update = False

class Connect_gui(tk.Frame):
    def __init__(self, parent):
        tk.Frame.__init__(self, parent, bg = "#FFF1C1", width = 400, height = 143)
        self.parent = parent
        self.history_opened = False
        self.frame = tk.Frame(master = self, bg = "#FFFDF4", width = 400, height =
74)
        self.frame.place(x = 0, y = 0)
        self.label1 = tk.Label(master = self, text = 'Адреса агента:',font =
("Tinos", 18), bg = '#FFFDF4', anchor='w')
        self.label1.place(x = 13, y = 5, width = 176, height = 28)
        self.entry_1 = tk.Entry(master = self, bd=0, bg='#FFFDF4',
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082')
        self.entry_1.place(x=13, y=34, width=374, height=37)
        self.button_image_1 =
tk.PhotoImage(file=assets.relative_to_assets("button_1.png"))
        self.button_1 = tk.Button(master = self, image=self.button_image_1,
text='Підключитись', font = ("Tinos", 18), compound='center', command =
self.button_1_handler, borderwidth=0, highlightthickness=0, relief="flat",
activebackground = "#FFF1C1")
        self.button_1.place(x=0, y=85, width=189, height=47)
        self.button_image_9 =
tk.PhotoImage(file=assets.relative_to_assets("button_9.png"))
        self.button_9 = tk.Button(master = self, command = self.history_button,
image=self.button_image_9, text='Історія', font = ("Tinos", 18), compound='center',
borderwidth=0, highlightthickness=0, relief="flat", activebackground = "#FFF1C1")
        self.button_9.place(x=211, y=85, width=189, height=47)

    def button_1_handler(self):
        agent = self.entry_1.get()
        if not agent in f.agents and agent != '':
            f.add_agent(agent, self)
        elif agent == '':
            tk.messagebox.showwarning('Увага!', 'Ви маєте ввести адресу!')
        elif not f.agents[agent].connection:
            f.connect(agent, self)

    def history_button(self):
        if not self.history_opened:
            self.history_window = history.History_window(self, self.entry_1.get())
            self.history_opened = True
        else:
            self.history_window.deiconify()
            self.history_window.attributes('-topmost', True)
            self.history_window.update()
            self.history_window.attributes('-topmost', False)
            self.history_window.focus()

class Graph_type(gui.Graph_type):
```

```
    def radio_button1(self):
        super().radio_button1()
        self.parent.label2['text'] = "Верхня допустима межа ентропії:"
        self.parent.label3['text'] = "Нижня допустима межа ентропії:"
        self.parent.entry_3.delete(0, 'end')
        self.parent.entry_3.insert(0,'0')
        self.parent.entry_4.delete(0, 'end')
        self.parent.entry_4.insert(0,'0')

    def radio_button2(self):
        super().radio_button2()
        self.parent.label2['text'] = "Значення ентропії за нормальної роботи:"
        self.parent.label3['text'] = "Допустиме відхилення:"
        self.parent.entry_3.delete(0, 'end')
        self.parent.entry_3.insert(0,'0')
        self.parent.entry_4.delete(0, 'end')
        self.parent.entry_4.insert(0,'0')

class Graph_params(tk.Frame):
    def __init__(self, parent):
        tk.Frame.__init__(self, parent, bg="#FFFDF4", width = 703, height = 333)
        self.label1 = tk.Label(master = self, text = 'Максимальна кількість подій на
графіку:',font = ("Tinos", 18), bg = '#FFFDF4', anchor='w')
        self.label1.place(x = 47, y = 22, width = 456, height = 28)
        self.label2 = tk.Label(master = self, text = 'Верхня допустима межа
ентропії:',font = ("Tinos", 18), bg = '#FFFDF4', anchor='w')
        self.label2.place(x = 47, y = 162, width = 456, height = 28)
        self.label3 = tk.Label(master = self, text = 'Нижня допустима межа
ентропії:',font = ("Tinos", 18), bg = '#FFFDF4', anchor='w')
        self.label3.place(x = 47, y = 208, width = 456, height = 28)
        self.graph_type = Graph_type(self)
        self.graph_type.place(x = 46, y = 67)
        self.button_image_2 =
tk.PhotoImage(file=assets.relative_to_assets("button_2.png"))
        self.button_2 = tk.Button(master = self, command = self.apply_graph_settings,
image=self.button_image_2, text='Підтвердити', font = ("Tinos", 18),
compound='center', borderwidth=0, highlightthickness=0, relief="flat",
activebackground = "#FFFDF4")
        self.button_2.place(x=444, y=264, width=213, height=47)
        self.button_image_5_1 =
tk.PhotoImage(file=assets.relative_to_assets("button_5_1.png"))
        self.button_image_5_2 =
tk.PhotoImage(file=assets.relative_to_assets("button_5_2.png"))
        self.button_image_5_3 =
tk.PhotoImage(file=assets.relative_to_assets("button_5_3.png"))
        self.button_5 = tk.Button(master = self, command = self.monitor_button,
image=self.button_image_5_1, text='Почати моніторинг', font = ("Tinos", 18),
compound='center', borderwidth=0, highlightthickness=0, relief="flat",
activebackground = "#FFFDF4")
        self.button_5.place(x=47, y=264, width=350, height=47)
        #max events on graph
        self.entry_2 = tk.Entry(master = self, bd=0, bg="#FFF1C1",
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082')
        self.entry_2.place(x=572, y=22, width=85, height=34)
        self.entry_2.insert(0,str(f.maxn))
        #normal enthropy
        self.entry_3 = tk.Entry(master = self, bd=0, bg="#FFF1C1",
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082')
        self.entry_3.place(x=572, y=162, width=85, height=34)
        self.entry_3.insert(0,'0')
        #violation
        self.entry_4 = tk.Entry(master = self, bd=0, bg="#FFF1C1",
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082')
        self.entry_4.place(x=572, y=208, width=85, height=34)
```

```
        self.entry_4.insert(0,'0')

    def apply_graph_settings(self):
        #global maxn, update, graph_mode, entry3v, entry4v, y, normal_entropy
        if self.entry_2.get() == '' and self.graph_type.button_graph_mode or
self.entry_3.get() == '' and not self.graph_type.button_graph_mode:
            tk.messagebox.showerror("Помилка!", "Ви маєте ввести параметри графіка!")
        else:
            try:
                entry2v = int(self.entry_2.get())
                entry3v = float(self.entry_3.get())
                entry4v = float(self.entry_4.get())
                if entry2v >= 5:
                    #read values
                    f.entry3v = entry3v
                    f.entry4v = entry4v
                    #graph mode
                    if self.graph_type.button_graph_mode: #1
                        f.graph_mode = True
                        f.normal_entropy = 0
                    else:                 #2
                        f.graph_mode = False
                        f.normal_entropy = f.entry3v
                    f.maxn = entry2v
                    #update y
                    f.change_events_on_graph()
                else:  tk.messagebox.showwarning("Увага!", "Кількість подій не має
бути меншою за 5.")
            except ValueError:
                tk.messagebox.showerror("Помилка!", "Ви маєте ввести число!")
        self.entry_2.delete(0, 'end')
        self.entry_2.insert(0,str(f.maxn))

    def monitor_button(self):
        if f.active_agent == '':
            tk.messagebox.showwarning("Увага!", "Підключення не виконано!")
        else:
            agent = f.agents[f.active_agent]
            if not agent.checkv and not agent.connection:
                tk.messagebox.showwarning("Увага!", "Підключення не виконано!")
            elif not agent.checkv and self.graph_type.button_graph_mode !=
f.graph_mode or not agent.checkv and f.entry3v != float(self.entry_3.get()) or not
agent.checkv and f.entry4v != float(self.entry_4.get()):
                tk.messagebox.showwarning("Увага!", "Ви не підтвердили зміни!")
            elif not agent.checkv and float(self.entry_3.get()) ==
float(self.entry_4.get()) and f.graph_mode:
                tk.messagebox.showwarning("Увага!", "Верхня і нижня межа не може
збігатись!")
            elif not agent.checkv and not agent.violation:
                self.button_5['image'] = self.button_image_5_2
                self.button_5['text'] = 'Перевищення не виявлено'
                agent.checkv = True
            elif not agent.violation:
                self.stop_monitoring(agent)
            else:
                self.button_5['image'] = self.button_image_5_2
                self.button_5['text'] = 'Перевищення не виявлено'
                agent.checkv = True
                agent.violation = False

    def stop_monitoring(self, agent):
        self.button_5['image'] = self.button_image_5_1
        self.button_5['text'] = 'Почати моніторинг'
        agent.checkv = False
```

```
class Message_gui(tk.Frame):
    def __init__(self, parent):
        super().__init__(parent)
        self.configure(bg = "#FFF1C1", width = 477, height = 323)
        self.button_message_method = True
        self.button_image_6_1 =
tk.PhotoImage(file=assets.relative_to_assets("button_6_1.png"))
        self.button_image_6_2 =
tk.PhotoImage(file=assets.relative_to_assets("button_6_2.png"))
        self.button_6 = tk.Button(master = self, command = self.email_button,
image=self.button_image_6_1, text='Email', font = ("Tinos", 18), compound='center',
borderwidth=0, highlightthickness=0, relief="flat", activebackground = "#FFF1C1")
        self.button_6.place(x=39, y=0, width=189, height=47)
        self.button_image_7_1 =
tk.PhotoImage(file=assets.relative_to_assets("button_7_1.png"))
        self.button_image_7_2 =
tk.PhotoImage(file=assets.relative_to_assets("button_7_2.png"))
        self.button_7 = tk.Button(master = self, command = self.telegram_button,
image=self.button_image_7_2, text='Telegram', font = ("Tinos", 18),
compound='center', borderwidth=0, highlightthickness=0, relief="flat",
activebackground = "#FFF1C1")
        self.button_7.place(x=251, y=0, width=188, height=47)
        self.frame = tk.Frame(master = self, bg = "#FFFDF4", width = 477, height =
276)
        self.frame.place(x = 0, y = 47)
        self.label6=tk.Label(master = self, text="Відправка повідомлень на пошту",
font=("Tinos", 18), bg="#FFFDF4")
        self.label6.place(x = 0, y = 56, height = 28, width = 477)
        #mail server
        self.label7=tk.Label(master = self, text="SMTP сервер та порт:",
font=("Tinos", 18), bg="#FFFDF4", anchor='w')
        self.label7.place(x = 22, y = 101, height = 28)
        self.entry_5 = tk.Entry(master = self, bd=0, bg="#FFF1C1",
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082')
        self.entry_5.place(x = 261, y = 99, width = 189, height = 34)
        #sender mail / chat id
        self.label8=tk.Label(master = self, text="Адреса відправника:",
font=("Tinos", 18), bg="#FFFDF4", anchor='w')
        self.label8.place(x = 22, y = 140, height = 28)
        self.entry_6 = tk.Entry(master = self, bd=0, bg="#FFF1C1",
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082')
        self.entry_6.place(x = 261, y = 138, width = 189, height = 34)
        #sender pssword
        self.label9=tk.Label(master = self, text="Пароль:", font=("Tinos", 18),
bg="#FFFDF4", anchor='w')
        self.label9.place(x = 22, y = 179, height = 28)
        self.entry_7 = tk.Entry(master = self, bd=0, bg="#FFF1C1",
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082', show="*")
        self.entry_7.place(x = 261, y = 177, width = 189, height = 34)
        #reciever email
        self.label10=tk.Label(master = self, text="Адреса отримувача:",
font=("Tinos", 18), bg="#FFFDF4", anchor='w')
        self.label10.place(x = 22, y = 221, height = 28)
        self.entry_8 = tk.Entry(master = self, bd=0, bg="#FFF1C1",
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082')
        self.entry_8.place(x = 261, y = 216, width = 189, height = 34)
        #apply_button
        self.button_image_8 =
tk.PhotoImage(file=assets.relative_to_assets("button_8.png"))
        self.button_8 = tk.Button(master = self, command = self.message_button,
image=self.button_image_8, text='Увімкнути повідомлення', font = ("Tinos", 18),
compound='center', borderwidth=0, highlightthickness=0, relief="flat",
activebackground = "#FFFDF4")
```

```
        self.button_8.place(x=98, y=265, width=305, height=47)

    def email_button(self):
        if not self.button_message_method:
            self.button_message_method=True
            self.label6['text'] = 'Відправка повідомлень на пошту'
            self.label7['text'] = 'SMTP сервер та порт:'
            self.label7.place(x = 22, y = 101)
            self.entry_5.place(x = 261, y = 99, width = 189)
            self.entry_5.delete(0, 'end')
            if f.server != '': self.entry_5.insert(0, f.server + ':' + str(f.port))
            else: self.entry_5.insert(0, '')
            self.label8['text'] = 'Адреса відправника:'
            self.label8.place(x = 22, y = 140)
            self.entry_6.place(x = 261, y = 138)
            self.entry_6.delete(0, 'end')
            self.entry_6.insert(0, f.sender_email)
            self.label9['text'] = 'Пароль:'
            self.label9.place(x = 22, y = 179)
            self.entry_7.place(x = 261, y = 177, width = 189, height = 34)
            self.entry_7.delete(0, 'end')
            self.entry_7.insert(0, f.password)
            self.label10['text'] = 'Адреса отримувача:'
            self.label10.place(x = 22, y = 218)
            self.entry_8.place(x = 261, y = 216, width = 189, height = 34)
            self.entry_8.delete(0, 'end')
            self.entry_8.insert(0, f.receiver_email)
            self.button_6['image'] = self.button_image_6_1
            self.button_7['image'] = self.button_image_7_2
            if f.mail_send: self.button_8['text'] = 'Вимкнути повідомлення'
            else: self.button_8['text'] = 'Увімкнути повідомлення'

    def telegram_button(self):
        if self.button_message_method:
            self.button_message_method=False
            self.label6['text'] = 'Відправка повідомлень в Telegram'
            self.label7['text'] = 'Введіть API token бота:'
            self.label7.place(x = 37, y = 108)
            self.entry_5.place(x = 37, y = 147, width = 404)
            self.entry_5.delete(0, 'end')
            self.entry_5.insert(0, f.token)
            self.label8['text'] = 'Введіть chat ID:'
            self.label8.place(x = 37, y = 196)
            self.entry_6.place(x = 252, y = 192)
            self.entry_6.delete(0, 'end')
            self.entry_6.insert(0, f.chat_id)
            self.label9['text'] = ''
            self.label9.place_forget()
            self.entry_7.place_forget()
            self.label10['text'] = ''
            self.label10.place_forget()
            self.entry_8.place_forget()
            self.button_6['image'] = self.button_image_6_2
            self.button_7['image'] = self.button_image_7_1
            if f.telegram_send: self.button_8['text'] = 'Вимкнути повідомлення'
            else: self.button_8['text'] = 'Увімкнути повідомлення'

    def message_button(self):
        try:
            if self.button_message_method and not f.mail_send:
                if self.entry_5.get() != '' and self.entry_6.get() != '' and
self.entry_7.get() != '' and self.entry_8.get() != '':
                    server = self.entry_5.get().split(':')
                    f.port = int(server[1])
```

```
                    if f.port != 465 and f.port != 587:
                        tk.messagebox.showwarning("Увага!", "Порт не відповідає
сервісу SMTP\nДоступні прти: 465, 587")
                    else:
                        f.server = server[0]
                        f.sender_email = self.entry_6.get()
                        f.password = self.entry_7.get()
                        f.receiver_email = self.entry_8.get()
                        f.email_try_to_login()
                        self.button_8['text'] = 'Вимкнути повідомлення'
                        f.mail_send = True
                else:
                    tk.messagebox.showerror("Помилка!", "Ви не ввели необхідні
дані!")
            elif self.button_message_method and f.mail_send:
                f.mail_send=False
                f.server = ''
                f.sender_email = ''
                f.password = ''
                f.receiver_email = ''
                self.button_8['text'] = 'Увімкнути повідомлення'
            elif not self.button_message_method and not f.telegram_send:
                if self.entry_5.get() != '' and self.entry_6.get() != '':
                    f.chat_id = self.entry_6.get()
                    f.token = self.entry_5.get()
                    self.button_8['text'] = 'Вимкнути повідомлення'
                    f.telegram_send = True
                else:
                    tk.messagebox.showerror("Помилка!", "Ви не ввели необхідні
дані!")
            elif not self.button_message_method and f.telegram_send:
                f.telegram_send = False
                f.chat_id = ''
                f.token = ''
                self.button_8['text'] = 'Увімкнути повідомлення'
        except IndexError:
            tk.messagebox.showwarning('Увага!', 'Ви маєте ввести SMTP порт!\nДоступні
прти: 465, 587')
        except Exception as e:
            tk.messagebox.showerror("Помилка!", e)

class main_gui(tk.Frame):
    def __init__(self, parent):
        super().__init__(parent)
        self.parent = parent
        self.parent.geometry("1200x700")
        self.parent.resizable(False, False)
        self.configure(bg = "#FFFCF4")
        #frames
        self.main_colorboxes = Main_colorboxes(self)
        self.main_colorboxes.place(x = 0, y = 0)
        self.connect_gui = Connect_gui(self)
        self.connect_gui.place(x = 752, y = 71)
        self.min_max_avg = gui.Min_max_avg(self)
        self.min_max_avg.place(x = 752, y = 214)
        self.graph_gui = Graph_gui(self)
        self.graph_gui.place(x = 0, y = 58)
        self.graph_params = Graph_params(self)
        self.graph_params.place(x = 0, y = 368)
        self.message_gui = Message_gui(self)
        self.message_gui.place(x = 713, y = 368)
        self.message_thread = f.Send_message_thread(self)
        self.message_thread.start()
```

```
    def close_connections(self):
        for connection in f.tcp_clients.values():
            connection.close_connection()
```

Аналізатор – файл «main_functions.py»

```
import socket, pickle, threading, io, requests, smtplib, ssl
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.image import MIMEImage
import matplotlib.pyplot as plt
import tkinter as tk
import queue
import db_functions as db

q=queue.Queue()

server = port = sender_email = password = receiver_email = chat_id = token = ''
mail_send = telegram_send = False
lock = threading.Lock()
agents = {}
tcp_clients = {}
active_agent = ''

normal_entropy = 0
maxn = 1000
entry3v = 0  #upper limit, normal_entropy
entry4v = 0  #lower limit, allowed_violation
graph_mode = True

def add_agent(addr, window):
    global active_agent
    agents[addr] = Agent()
    active_agent = addr
    connect(addr, window)

def connect(addr, window):
    lock.acquire()
    tcp_clients[addr] = Tcp_client_thread(addr, window)
    if not agents[addr].connection:
        agents[addr].y.clear()
        tcp_clients[addr].start()
        agents[addr].connection = True
    else:
        tk.messagebox.showwarning("Увага!", "Підключення вже виконано.")
    lock.release()

def change_events_on_graph():
    if active_agent != '':
        agent = agents[active_agent]
        with agent.varlock:
            agent.y.clear()
            for event in db.get_n_last_events(maxn, active_agent):
                agent.y.insert(0, abs(event[1] - normal_entropy))
            agent.update = True

def email_try_to_login():
    context = ssl.create_default_context()
    if port == 587:
        mail = smtplib.SMTP(server,port)
```

```
        mail.ehlo()
        mail.starttls(context=context)
        mail.ehlo()
        mail.login(sender_email, password)
        mail.quit()
    else:
        with smtplib.SMTP_SSL(server, port, context=context) as mail:
            mail.login(sender_email, password)

class Agent():
    def __init__(self):
        self.update = False
        self.connection = False
        self.y = []
        self.checkv = False
        self.violation = False
        self.vi_val = 0   #violation value
        self.varlock = threading.Lock()

class Tcp_client_thread(threading.Thread):
    def __init__(self, addr, window):
         threading.Thread.__init__(self, daemon=True)
         self.addr = addr
         self.window = window

    def run(self):
        agents[self.addr].connection = True
        if self.addr == active_agent: self.window.button_1['text'] = 'Підключення...'
        address = (self.addr,1234)
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        try:
            self.s.connect(address)
        except Exception as e:
            tk.messagebox.showerror("Помилка!", e) #"IP ареса неправильна!"
            if self.addr == active_agent: self.window.button_1['text'] =
'Підключитись'
            agents[self.addr].connection = False
        if agents[self.addr].connection:
            if self.addr == active_agent: self.window.button_1['text'] = 'Підключено'
            try:
                db.create_db_table(self.addr)
                print('connected')
                while True:
                    data = self.s.recv(66)
                    event = pickle.loads(data)
                    self.add_new_value(event)
            except Exception as e:
                self.s.close()
                if self.addr == active_agent: self.window.button_1['text'] =
'Підключитись'
                self.window.parent.graph_params.stop_monitoring(agents[self.addr])
                agents[self.addr].connection = False
                tk.messagebox.showwarning("Увага!", e) #"Підключення закрито."

    def add_new_value(self, event):
        val = abs(event[1] - normal_entropy)
        with agents[self.addr].varlock:
            agents[self.addr].y.append(val)
        db.update_db(self.addr, event)
        self.check(val)
        with agents[self.addr].varlock:
            if len(agents[self.addr].y) > maxn:
                agents[self.addr].y.pop(0)
            agents[self.addr].update = True
```

```
    def check(self, val):
        if val > entry3v and graph_mode and agents[self.addr].checkv or val < entry4v
and graph_mode and agents[self.addr].checkv or val > entry4v and not graph_mode and
agents[self.addr].checkv:
            tk.messagebox.showwarning("Увага!", "Перевищення!")
            with agents[self.addr].varlock:
                agents[self.addr].vi_val = val
                agents[self.addr].violation = True
                agents[self.addr].checkv = False
            q.put(self.addr)

    def close_connection(self):
        self.s.close()

class Send_message_thread(threading.Thread):
    def __init__(self, window):
        threading.Thread.__init__(self, daemon=True)
        self.window = window

    def run(self):
        while True:
            addr = q.get()
            self.agent=agents[addr]
            if addr == active_agent:
                self.window.graph_params.button_5['image'] =
self.window.graph_params.button_image_5_3
                self.window.graph_params.button_5['text'] = 'Перевищення!'
            im = self.create_image()
            if telegram_send: self.telegram_send_message(im)
            if mail_send: self.email_send_message(im)
            q.task_done()

    def create_image(self):
        with self.agent.varlock:
            y = self.agent.y
            fig = plt.Figure(figsize=(7,3),dpi=200)
            fig.subplots_adjust(left=0.07, right=0.96, top=0.96, bottom=0.11)
            x = range(1, len(y) + 1)
            a = fig.add_subplot()
            a.fill_between(x, y, alpha = 0.2, color = "#CFA000")
            a.plot(x, y, color = "#CFA000")
            a.set_ylim(bottom=0)
            ymin, ymax = a.get_ybound()
            a.set_ybound((ymin, ymax*1.2))
            buf = io.BytesIO()
            fig.savefig(buf, format='png')
            buf.seek(0)
            return buf

    def telegram_send_message(self, im):
        action = "sendPhoto"
        text = self.get_text(True)
        data = {"chat_id": chat_id, "caption": text}
        requests.post("https://api.telegram.org/bot{0}/".format(token) + action,
data=data , files={"photo": im})

    def email_send_message(self, im):
        context = ssl.create_default_context()
        message = self.email_form_message(im)
        if port == 587:
            try:
                mail = smtplib.SMTP(server,port)
                mail.ehlo()
```

```
                mail.starttls(context=context)
                mail.ehlo()
                mail.login(sender_email, password)
                mail.sendmail(sender_email, receiver_email, message.as_string())
            except Exception as e:
                tk.messagebox.showerror("Помилка!", e)
            finally:
                mail.quit()
        else:
            try:
                with smtplib.SMTP_SSL(server, port, context=context) as mail:
                    mail.login(sender_email, password)
                    mail.sendmail(sender_email, receiver_email, message.as_string())
            except Exception as e:
                tk.messagebox.showerror("Помилка!", e)

    def email_form_message(self, im):
        message = MIMEMultipart("alternative")
        message["Subject"] = "Моніторинг ентропії"
        message["From"] = sender_email
        message["To"] = receiver_email
        message.attach(MIMEText(self.get_text(False), "plain"))
        message.attach(MIMEText(self.get_html(), "html"))
        msgImage = MIMEImage(im.getvalue())
        msgImage.add_header('Content-ID', '<image>')
        message.attach(msgImage)
        return message

    def get_text(self, mode): #true - emoji, false - no emoji
        agent = agents[active_agent]                    #--------------------only-one-
agent
        if mode:
            if graph_mode:
                text = '''⚠ Перевищення ⚠\n
☑ Режим графіка: зміна ентропії\n
🔘 Поточне значення ентропії: {0}\n
✅ Допустимий діапазон: {1} - {2}'''.format(agent.vi_val, entry4v, entry3v)
            else:
                text = '''⚠ Перевищення ⚠\n
☑ Режим графіка: відхилення ентропії\n
🔘 Поточне значення: {0}\n
✅ Допустиме відхилення: {1}'''.format(agent.vi_val, entry4v)
        else:
            if graph_mode:
                text = '''Перевищення!\n
Режим графіка: зміна ентропії\n
Поточне значення ентропії: {0}\n
Допустимий діапазон: {1} - {2}'''.format(agent.vi_val, entry4v, entry3v)
            else:
                text = '''Перевищення!\n
Режим графіка: відхилення ентропії\n
Поточне значення: {0}\n
Допустиме відхилення: {1}'''.format(agent.vi_val, entry4v)
        return text

    def get_html(self):
        agent = agents[active_agent]                    #--------------------only-
one-agent
        if graph_mode:
            html = '''\
<html>
<body>
```

```
    <table style="background-color:#FFF1C1" cellpadding="12">
    <tr>
    <td>
    <table style="background-color:#FFFDF4" cellpadding="12">
    <tr>
    <td>
    <br><img src="cid:image" width="700" height="300"><br>
    <p style="font-size:24px">⚠ Перевищення ⚠</p>
    <p style="font-size:24px">☑ Режим графіка: зміна ентропії</p>
    <p style="font-size:24px">🔘 Поточне значення ентропії: {0}</p>
    <p style="font-size:24px">✅ Допустимий діапазон: {1} - {2}</p>
    </td>
    </tr>
    </table>
    </tr>
    </td>
    </table>
</body>
</html>
    '''.format(agent.vi_val, entry4v, entry3v)
        else:
            html = '''\
<html>
<body>
    <table style="background-color:#FFF1C1" cellpadding="12">
    <tr>
    <td>
    <table style="background-color:#FFFDF4" cellpadding="12">
    <tr>
    <td>
    <br><img src="cid:image" width="700" height="300"><br>
    <p style="font-size:24px">⚠ Перевищення ⚠</p>
    <p style="font-size:24px">☑ Режим графіка: відхилення ентропії</p>
    <p style="font-size:24px">🔘 Поточне значення: {0}</p>
    <p style="font-size:24px">✅ Допустиме відхилення: {1}</p>
    </td>
    </tr>
    </table>
    </tr>
    </td>
    </table>
</body>
</html>
    '''.format(agent.vi_val, entry4v)
        return html
```

Аналізатор – файл «history_window.py»

```
import tkinter as tk
from tkinter import ttk
from tkcalendar import DateEntry
import shared_gui as gui
import db_functions as db
import relative_to_assets as assets
from datetime import time, datetime
import matplotlib.dates as mdates
from matplotlib import pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MaxNLocator
```

```
class Main_colorboxes(tk.Frame):
    def __init__(self,parent):
        tk.Frame.__init__(self, parent, bg = "#FFDEAD", height = 58, width = 1200)
        self.parent = parent
        self.label1 = tk.Label(master = self.parent, text = 'Історія значень
ентропії',font = ("Tinos", 24), bg = '#FFDEAD',fg="#4B0082")
        self.label1.place(x = 0, y = 12, width = 1200, height = 36)

class DateTime_entry(tk.Frame):
    def __init__(self,parent):
        tk.Frame.__init__(self, parent, bg = "#FFFDF4", width = 310, height = 34)
        self.style = ttk.Style(self)
        self.style.theme_use('clam')
        self.style.configure('DateEntry', fieldbackground='#FFF1C1',
foreground="#4B0082", bordercolor = '#FFF1C1', lightcolor = '#FFF1C1', darkcolor =
'#FFF1C1')
        self.style.configure('TSpinbox', fieldbackground='#FFF1C1',
foreground="#4B0082", arrowsize=15, bordercolor = '#FFF1C1', lightcolor = '#FFF1C1',
darkcolor = '#FFF1C1')
        self.date_entry = DateEntry(self, selectmode='day', style='DateEntry', font =
("Tinos", 16), locale='uk_UA', date_pattern = 'dd.MM.yyyy',
        headersbackground='#FFF1C1', normalbackground = '#FFFDF4', bordercolor =
'#E8CD70', background = '#E8CD70', foreground = 'black')
        self.date_entry.place(x=0, y=0, width=136, height=34)
        self.hour_string = tk.StringVar(value=0)
        self.min_string = tk.StringVar(value=0)
        self.sec_string = tk.StringVar(value=0)
        self.hour_enty = ttk.Spinbox(master = self, from_=0, to=23, wrap=True,
style='TSpinbox', textvariable=self.hour_string, font = ("Tinos", 16))
        self.hour_enty.place(x=138, y=0, width=55, height=34)
        self.min_entry = ttk.Spinbox(master = self, from_=0, to=59, wrap=True,
style='TSpinbox', textvariable=self.min_string, font = ("Tinos", 16))
        self.min_entry.place(x=195, y=0, width=55, height=34)
        self.sec_entry = ttk.Spinbox(master = self, from_=0, to=59,wrap=True,
style='TSpinbox', textvariable=self.sec_string, font = ("Tinos", 16))
        self.sec_entry.place(x=252, y=0, width=55, height=34)

    def get_time(self):
        h = self.hour_enty.get()
        m = self.min_entry.get()
        s = self.sec_entry.get()
        timev = time(hour=int(h), minute=int(m), second=int(s))
        return timev

class History_params(tk.Frame):
    def __init__(self,parent):
        tk.Frame.__init__(self, parent, bg = "#FFFDF4", width = 440, height = 205)
        self.label1 = tk.Label(master = self, text = 'Адреса агента:',font =
("Tinos", 18), bg = '#FFFDF4', anchor='w')
        self.label1.place(x = 13, y = 15, width = 166, height = 28)
        self.label2 = tk.Label(master = self, text = 'Часовий діапазон:',font =
("Tinos", 18), bg = '#FFFDF4')
        self.label2.place(x = 0, y = 53, width = 440, height = 28)
        self.label3 = tk.Label(master = self, text = 'Початок:',font = ("Tinos", 18),
bg = '#FFFDF4', anchor='w')
        self.label3.place(x = 13, y = 121, width = 103, height = 28)
        self.label3 = tk.Label(master = self, text = 'Кінець:',font = ("Tinos", 18),
bg = '#FFFDF4', anchor='w')
        self.label3.place(x = 13, y = 163, width = 87, height = 28)
        self.label4 = tk.Label(master = self, text = 'Дата',font = ("Tinos", 14), bg
= '#FFFDF4')
        self.label4.place(x = 116, y = 88, width = 136, height = 24)
        self.label5 = tk.Label(master = self, text = 'год',font = ("Tinos", 14), bg =
'#FFFDF4')
```

```
        self.label5.place(x = 254, y = 88, width = 55, height = 24)
        self.label6 = tk.Label(master = self, text = 'хв',font = ("Tinos", 14), bg =
'#FFFDF4')
        self.label6.place(x = 311, y = 88, width = 55, height = 24)
        self.label7 = tk.Label(master = self, text = 'сек',font = ("Tinos", 14), bg =
'#FFFDF4')
        self.label7.place(x = 368, y = 88, width = 55, height = 24)
        self.entry_1 = tk.Entry(master = self, bd=0, bg='#FFF1C1',
highlightthickness=0, font = ("Tinos", 18), fg = '#4B0082')
        self.entry_1.place(x = 179, y = 12, width = 244, height = 34)
        self.datetime_entry_1 = DateTime_entry(self)
        self.datetime_entry_1.place(x = 116, y = 116)
        self.datetime_entry_2 = DateTime_entry(self)
        self.datetime_entry_2.place(x = 116, y = 158)

class No_data_label(tk.Frame):
    def __init__(self,parent):
        tk.Frame.__init__(self, parent, bg = "#D4D4D4", width = 318, height = 57)
        self.label1 = tk.Label(master = self, text = 'Немає даних',font = ("Tinos",
18), bg = 'white', borderwidth=3, relief='solid')
        self.label1 = tk.Label(master = self, text = 'Немає даних',font = ("Tinos",
18), bg = 'white')
        self.label1.place(x = 2, y = 2, width = 314, height = 53)

class Graph_gui(gui.Graph_gui):
    def __init__(self, parent):
        super().__init__(parent)
        self.graph.place(x = 4, y = 4, width = 738, height = 388)
        self.canvas_graph.get_tk_widget().config(width=738,height=388)

    def show_graph_n(self, y):
        self.fig.subplots_adjust(bottom=0.075)
        self.plot_graph(y)
        self.canvas_graph.draw()
        self.show_min_max_avg(self.parent, y)

    def show_graph_date(self, x, y, t1, t2):
        self.a.clear()
        self.fig.subplots_adjust(bottom=0.3)
        self.a.plot(x, y, '.', markersize=1, color = "#CFA000")
        self.a.minorticks_on()
        self.set_y_axis_params()
        self.set_x_axis_date_params(t1, t2)
        self.canvas_graph.draw()
        self.show_min_max_avg(self.parent, y)

    def set_x_axis_date_params(self, t1, t2):
        self.a.xaxis.set_major_formatter(mdates.DateFormatter('%d.%m.%y %H:%M'))
        self.a.xaxis.set_minor_formatter(mdates.DateFormatter('%H:%M:%S'))
        self.a.xaxis.set_minor_locator(AutoMinorLocator())
        n = (t2.replace(hour=0, minute=0, second=0) - t1.replace(hour=0, minute=0,
second=0)).days
        if n<5 and n !=0:
            self.a.xaxis.set_major_locator(MaxNLocator(n))
        elif n==0:
            self.a.xaxis.set_major_locator(MaxNLocator(1))
        else:
            self.a.xaxis.set_major_locator(MaxNLocator(5))
        plt.setp(self.a.get_xticklabels(minor = True) , rotation=90,
horizontalalignment='right', fontsize=11)
        plt.setp(self.a.get_xticklabels(), rotation=90, horizontalalignment='right',
fontsize=11)
        self.a.set_xlim(left = t1, right = t2)
```

```
class Graph_type(gui.Graph_type):
    def __init__(self, parent):
        super().__init__(parent)
        self.configure(width = 716)
        self.label5['text'] = 'Ентропія від кількості подій'
        self.label6['text'] = 'Ентропія від дати/часу'
        self.button_3.place(x = 27, y = 44)
        self.button_4.place(x = 402, y = 44)
        self.label5.place(x = 64, y = 42, width = 315, height = 28)
        self.label6.place(x = 439, y = 42, width = 258, height = 28)

class History_window(tk.Toplevel):
    def __init__(self, parent, addr):
        super().__init__(parent, bg = "#FFF1C1", width = 1200, height = 569)
        self.parent = parent
        self.resizable(False, False)
        self.protocol("WM_DELETE_WINDOW", self.on_closing)
        self.main_colorboxes = Main_colorboxes(self)
        self.main_colorboxes.place(x = 0, y = 0)
        self.graph_gui = Graph_gui(self)
        self.graph_gui.place(x = 0, y = 58, width = 746, height = 396)
        self.history_params = History_params(self)
        self.history_params.entry_1.insert(0,addr)
        self.history_params.place(x = 753, y = 65)
        self.button_image_10 =
tk.PhotoImage(file=assets.relative_to_assets("button_10.png"))
        self.button_10 = tk.Button(master = self, command = self.show,
image=self.button_image_10, text = 'Показати графік', font = ("Tinos", 18), compound
= 'center', borderwidth=0, highlightthickness=0, relief='flat', activebackground =
"#FFF1C1", background = "#FFF1C1")
        self.button_10.place(x = 860, y = 277)
        self.min_max_avg = gui.Min_max_avg(self)
        self.min_max_avg.place(x = 773, y = 331)
        self.frame = tk.Frame(master = self, bg = "#FFFDF4", width = 746, height =
115)
        self.frame.place(x = 0, y = 454)
        self.graph_type = Graph_type(self)
        self.graph_type.place(x = 15, y = 470)
        self.no_data_label = No_data_label(self)
        self.no_data_label.place(x = 214, y = 228)

    def show(self):
        try:
            date_1 = self.history_params.datetime_entry_1.date_entry.get_date()
            date_2 = self.history_params.datetime_entry_2.date_entry.get_date()
            time1 = self.history_params.datetime_entry_1.get_time()
            time2 = self.history_params.datetime_entry_2.get_time()
            datetime1 = datetime.combine(date_1, time1)
            datetime2 = datetime.combine(date_2, time2)
            addr = self.history_params.entry_1.get()
            if db.check_if_table_exists(addr):
                events = db.get_events_by_date([datetime1.strftime('%Y-%m-%d
%H:%M:%S'), datetime2.strftime('%Y-%m-%d %H:%M:%S')], addr)
                if len(events) > 1:
                    self.no_data_label.place_forget()
                    y = []
                    for event in events:
                        y.append(event[1])
                    if self.graph_type.button_graph_mode:
                        self.graph_gui.show_graph_n(y)
                    else:
                        x = []
                        for event in events:
```

```
                            x.append(datetime.strptime(event[0], '%Y-%m-%d
%H:%M:%S.%f'))
                        self.graph_gui.show_graph_date(x, y, datetime1, datetime2)
                else: self.no_data()
            else: self.no_data()
        except:
            tk.messagebox.showerror("Помилка!", "Введено неправильні дані!",
parent=self)

    def no_data(self):
        self.graph_gui.clear_graph()
        self.graph_gui.set_axis_params()
        self.graph_gui.canvas_graph.draw()
        self.graph_gui.show_min_max_avg(self, [0, 0])
        self.no_data_label.place(x = 214, y = 228)

    def on_closing(self):
        self.parent.history_opened = False
        self.destroy()
```

Аналізатор – файл «db_functions.py»

```
import sqlite3
import threading
db = sqlite3.connect('events.db', check_same_thread=False)
lock = threading.Lock()

def get_n_last_events(n, addr):
    lock.acquire()
    c = db.cursor()
    sql='''SELECT * FROM \'{0}\' ORDER BY rowid DESC LIMIT {1}'''.format(addr, n)
    c.execute(sql)
    events = c.fetchall()
    lock.release()
    return events

def get_events_by_date(date_range, addr):
    lock.acquire()
    c = db.cursor()
    sql='''SELECT * FROM \'{0}\' WHERE EventDateTime >= \'{1}\' AND EventDateTime <
\'{2}\' ORDER BY rowid'''.format(addr, date_range[0], date_range[1])
    c.execute(sql)
    events = c.fetchall()
    lock.release()
    return events

def check_if_table_exists(addr):
    lock.acquire()
    c = db.cursor()
    c.execute('''SELECT count(name) FROM sqlite_master WHERE type='table' AND
name=\'{0}\'''').format(addr))
    if c.fetchone()[0] == 0: table = False
    else: table = True
    c.close()
    lock.release()
    return table

def create_db_table(addr):
    if not check_if_table_exists(addr):
        lock.acquire()
```

```
        db.execute('''CREATE TABLE \'{0}\' (
        EventDateTime timestamp,
        Entropy float);'''.format(addr))
        lock.release()

def update_db(addr, event):
    lock.acquire()
    sql = '''INSERT INTO \'{0}\' (EventDateTime, Entropy)
    VALUES (?,?);'''.format(addr)
    values = (event[0],event[1])
    db.execute(sql, values)
    db.commit()
    lock.release()
```

Аналізатор – файл «shared_gui.py»

```
import tkinter as tk
from matplotlib import pyplot as plt
from matplotlib import style
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.ticker import MaxNLocator
import relative_to_assets as assets

class Graph_gui(tk.Frame):
    def __init__(self, parent):
        tk.Frame.__init__(self, parent, bg = "#E8CD70", width = 703, height = 310)
        self.parent = parent
        style.use('seaborn-whitegrid')
        self.graph = tk.Frame(relief=tk.FLAT, borderwidth=0, master = self)
        self.graph.place(x = 4, y = 4, width = 695, height = 302)
        self.fig = plt.Figure(figsize=(7,3),dpi=100)
        self.fig.subplots_adjust(left=0.07, right=0.96, top=0.96, bottom=0.11)
        self.fig.patch.set_facecolor('#FFFDF4')
        self.a = self.fig.add_subplot()
        self.a.tick_params(axis='x', labelsize=14)
        self.a.tick_params(axis='y', labelsize=14)
        self.set_axis_params()
        self.canvas_graph = FigureCanvasTkAgg(self.fig, self.graph)
        self.canvas_graph.draw()
        self.canvas_graph.get_tk_widget().pack()
        self.canvas_graph.get_tk_widget().config(width=703,height=310)

    def clear_graph(self):
        self.a.clear()

    def set_axis_params(self):
        self.set_y_axis_params()
        self.a.set_xlim(left=0)
        self.a.margins(x=0)
        self.a.xaxis.set_major_locator(MaxNLocator(integer=True))

    def set_y_axis_params(self):
        self.a.set_ylim(bottom=0)
        self.ymin, self.ymax = self.a.get_ybound()
        self.a.set_ybound((self.ymin, self.ymax*1.2))

    def plot_graph(self, y):
        self.a.clear()
        x=range(1, len(y) + 1)
        self.a.fill_between(x, y, alpha = 0.2, color = "#CFA000")
```

```
        self.a.plot(x, y, color = "#CFA000")
        self.set_axis_params()

    def show_min_max_avg(self,target,y):
        avgv = '{:.2f}'.format(sum(y)/len(y),2)
        minv = '{:.2f}'.format(min(y),2)
        maxv = '{:.2f}'.format(max(y),2)
        target.min_max_avg.set_values(minv, maxv, avgv)

class Min_max_avg(tk.Frame):
    def __init__(self, parent, *args, **kwargs):
        tk.Frame.__init__(self, parent, bg = "#FFFDF4", width = 400, height = 142)
        self.label1 = tk.Label(master = self, text = '0.00',font = ("Tinos", 18), bg
= '#FFF1C1', anchor='s')
        self.label1.place(x = 290, y = 12, width = 85, height = 34)
        self.label2 = tk.Label(master = self, text = '0.00',font = ("Tinos", 18), bg
= '#FFF1C1', anchor='s')
        self.label2.place(x = 290, y = 54, width = 85, height = 34)
        self.label3 = tk.Label(master = self, text = '0.00',font = ("Tinos", 18), bg
= '#FFF1C1', anchor='s')
        self.label3.place(x = 290, y = 96, width = 85, height = 34)
        self.label4 = tk.Label(master = self, text = 'Максимальне значення:',font =
("Tinos", 18), bg = '#FFFDF4', anchor='w')
        self.label4.place(x = 18, y = 98, width = 269, height = 28)
        self.label5 = tk.Label(master = self, text = 'Середнє значення:',font =
("Tinos", 18), bg = '#FFFDF4', anchor='w')
        self.label5.place(x = 18, y = 15, width = 269, height = 28)
        self.label6 = tk.Label(master = self, text = 'Мінімальне значення:',font =
("Tinos", 18), bg = '#FFFDF4', anchor='w')
        self.label6.place(x = 18, y = 57, width = 269, height = 28)

    def set_values(self, minv, maxv, avgv):
        self.label1['text'] = avgv
        self.label2['text'] = minv
        self.label3['text'] = maxv

class Graph_type(tk.Frame):
    def __init__(self, parent, *args, **kwargs):
        tk.Frame.__init__(self, parent, bg = "#FFF1C1", width = 611, height = 83)
        self.button_graph_mode = True
        self.parent = parent
        self.label4 = tk.Label(master = self, text = 'Режим графіка:',font =
("Tinos", 18), bg = '#FFF1C1', anchor='w')
        self.label4.place(x = 12, y = 3, width = 172, height = 28)
        self.label5 = tk.Label(master = self, text = 'Зміна ентропії',font =
("Tinos", 18), bg = '#FFF1C1', anchor='w')
        self.label5.place(x = 71, y = 42, width = 174, height = 28)
        self.label6 = tk.Label(master = self, text = 'Відхилення ентропії',font =
("Tinos", 18), bg = '#FFF1C1', anchor='w')
        self.label6.place(x = 325, y = 42, width = 229, height = 28)
        self.button_image_3 =
tk.PhotoImage(file=assets.relative_to_assets("button_3.png"))
        self.button_3 = tk.Button(master = self, command = self.radio_button1,
image=self.button_image_3, borderwidth=0, highlightthickness=0, relief="flat",
activebackground = "#FFF1C1")
        self.button_3.place(x=34, y=44, width=26, height=26)
        self.button_image_4 =
tk.PhotoImage(file=assets.relative_to_assets("button_4.png"))
        self.button_4 = tk.Button(master = self, command = self.radio_button2,
image=self.button_image_4, borderwidth=0, highlightthickness=0, relief="flat",
activebackground = "#FFF1C1")
        self.button_4.place(x=288, y=44, width=26, height=26)

    def radio_button1(self):
```

```
        self.button_3['image'] = self.button_image_3
        self.button_4['image'] = self.button_image_4
        self.button_graph_mode = True

    def radio_button2(self):
        self.button_3['image'] = self.button_image_4
        self.button_4['image'] = self.button_image_3
        self.button_graph_mode = False
```

Аналізатор – файл «relative_to_assets.py»

```
from pathlib import Path

OUTPUT_PATH = Path(__file__).parent
ASSETS_PATH = OUTPUT_PATH / Path("./assets")

def relative_to_assets(path: str) -> Path:
    return ASSETS_PATH / Path(path)
```