Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

Спеціальність 125 «Кібербезпека»

Освітня програма «Безпека інформаційних та комунікаційних систем»

«Допущено до захисту»

В.о. зав.кафедрою БІСТ

Ольга

МЕЛКОЗЬОРОВА

_____

« » 2023р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Аналіз технологій спостереження і екстракції даних про мережеву активність користувачів та синтез алгоритму компіляції одиничних «демон-юнітів» с заданими поведінковими властивостями»

оцінка « »

Голова ЕК

Олександр ЛЕМЕШКО _____

Керівник доцент каф. БІСТ

Малахов С.В._____

Рецензент Головний метролог ХНУ імені Каразіна _____

Гостєв О.Л._____

Виконавець : студент групи КБ-61

Азаров Сергій Ігорович _____

Харків – 2023

РЕФЕРАТ

The explanatory note contains 67 pages, 8 figures, 1 appendix, and 17 sources.

The purpose of the thesis is to identify and study the typical behavioral characteristics of users of modern information systems and to synthesize an algorithm for compiling single "daemon units" with specified properties.

The object of research of the thesis is methods of network monitoring and active protection of information resources of modern IS.

The subject of research is the procedures for interception, extraction and emulation of network interaction processes.

Main research methods: - computer modeling and generalization of the results.

To achieve this goal, the method of analysis and synthesis was used.

Based on the results of the analysis of known methods of interception, extraction and generalization of behavioral data on network activity of network users, a software implementation for compiling single "daemon units" was created.

The investigated procedures for monitoring and data extraction provide the necessary input data for the algorithm for compiling single "daemon units" with specified behavioral properties (profiles) of network users.

The software model of the experimental algorithm for the synthesis of behavioral daemon units is implemented in C++. UML modeling was used for software development. The results of the work are an integral part of the research, within the framework of the general concept of creating a prototype algorithm for automatic emulation of network presence for single "daemon units" with specified parameters of behavioral profiles.

Keywords: UNIT, DAEMON, MODULE, PROCESS, IB, UML, ALGORITHM, CONFIGURATION, NETWORK PROFILE

ABSTRACT

Master's Qualification Work: 67 pages, 8 figures, 15 sources. The work includes an introduction, six chapters, conclusions, and a list of references.

The aim of the thesis is to research the synthesis of the algorithm for compiling single 'daemon units' that can effectively monitor anomalies in user behavior and potential threats in the network and take the necessary actions to protect against unauthorized access.

The object of research of the thesis is the process of network surveillance, interception, as well as data extraction for the purpose of detecting and preventing threats.

The subject of development is an algorithm for automated compilation of daemon units

Research Methods: In order to succeed in this research, I used the analysis and synthesis method. All available information related to this problem was taken into account. As a result of the analysis of technologies and tools, a proprietary compilation algorithm was proposed.

The software is implemented in the C++ language. Modeling using UML was used for software development.

The results of the work are an algorithm for the automated compilation of daemon units

Keywords: UNIT, BLOCK, DEMON, MODULE, PROCESS, ALGORITHM, CONFIGURATION

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС - інформаційна система.

КІ - користувальницький інтерфейс

IDS - система виявлення вторгнень.

IPS - система запобігання вторгнень.

SNMP - протокол управління мережею.

ВСТУП

У сучасну цифрову епоху, коли величезні масиви даних обробляються в режимі реального часу, а мережева активність стає невід'ємною частиною повсякденного життя сучасної людини, виникає потреба в глибокому аналізі та ефективному управлінні цими процесами. Дана робота спрямована на дослідження процесів та розробку програмних інструментів, які забезпечують можливості адміністрування мережевого трафіку та відтворення потрібних поведінкових властивостей мережевих користувачів, що, в свою чергу, створює необхідну основу для подальшого синтезу спеціалізованих безпекових платформ, що віртуалізують задані процеси та/або елементи мережевої інфраструктури сучасних інформаційних систем (ІС).

Зі зростанням кількості підключених пристроїв та розвитком хмарних інфраструктур мережева активність стає все більш складною та різноманітною. Розуміння цієї екосистеми та взаємодії її структурних елементів важливо не тільки для підвищення ефективності, але й для вирішення задач безпеки цієї діяльності. В цьому контексті, процес аналізу відомих технологій мережевого спостереження є ключовим аспектом для успішного вирішення завдань на наступних етапах розробки систем емуляції мережевих співтовариств (наприклад, створення т.з. «бот-ферм» та/чи «бот- екосистем» із заданими властивостями, котрі орієнтовані для відтворення різних функціональних завдань).

Вочевидь, що ретельний аналіз існуючих технологій перехоплення, моніторингу та узагальнення даних про мережеву активність користувачів сучасних ІС, є своєрідним фундаментом для подальшого синтезу спеціалізованих алгоритмів компіляції одиничних або групових «демон-юнітів» із заданими поведінковими властивостями. Вирішення цих завдань дозволить підвищити якість мережевого аналізу та забезпечити більш точніше виявлення аномалій та потенційних загроз безпеки. І це тільки на першому рівні загального циклу

досліджень, що є вихідним етапом для вирішення більш складних завдань, щодо створення VR- екосистем.

На сьогоднішній день існує безліч методів і методик моніторингу мережевої активності, але багато з них стикаються з труднощами при виявленні та подальшої інтерпретації складних патернів поведінки мережевих користувачів. Очевидно, що від ступеня невизначеності вилучених користувальницьких патернів залежить результат роботи (тобто адекватність) емулюючого ядра, навіть на рівні синтезу одиночних демон-юнітів. При цьому перехід на групову обробку або спроба повної автоматизації процесу синтезу одиничного демон-юніту стає складно формалізованим завданням, ступінь складності якого знаходиться в прямій залежності з адекватністю та повнотою вилучених мережевих даних (наприклад - процеси збору інформації про параметри роботи смартфонів і встановлених в них додатків, що реалізовані (за замовчуванням) у програмних уставках від компанії Google (ОС Android-Налаштування-Сервіси Google і далі за налаштуваннями...)). Існуючи технології не завжди враховують весь спектр можливих сценаріїв розвитку подій, що вказує на необхідність подальших досліджень та впровадження інноваційних рішень у цій галузі. Саме тому представлена дипломна робота сфокусована на розгляді питань, що охоплюють дослідження існуючих технологій моніторингу мережевої активності та аналізу методів й інструментів вилучення потрібних даних, що уможливлює формування потрібних умов для розробки алгоритму компіляції одиночного демон-юниту з тестовим набором поведінкових характеристик. Кожен розділ роботи спрямований на вирішення певної частини завдань дослідження та досягнення загальної мети - визначення й дослідження типових поведінкових особливостей користувачів сучасних інформаційних систем.

1 ОГЛЯД І УЗАГАЛЬНЕННЯ ОСНОВНИХ РІЗНОВИДІВ ТА ХАРАКТЕРИСТИК МЕРЕЖЕВОЇ АКТИВНОСТІ ТИПОВИХ СТРУКТУРНИХ ЕЛЕМЕНТІВ СУЧАСНИХ ІНФОРМАЦІЙНИХ СИСТЕМ ТА СЕРВІСІВ

1.1 Аналіз особливостей функціонування основних різновидів мережевих додатків та різних категорій користувачів сучасних ІС

Мережева діяльність - це процес обміну даними, інформацією або ресурсами між різними пристроями, такими як комп'ютери або сервери, та користувачами через комп'ютерні мережі. Ця діяльність відрізняється за обсягом даних, швидкістю передачі, напрямком зв'язку та іншими характеристиками. Комп'ютерні мережі можна класифікувати за різними критеріями, такими як масштаб, фізична топологія, архітектура та призначення. Розглянемо основні типи мереж за різними параметрами:

Масштаб:

- Локальна мережа (LAN): Обмежена територія, зазвичай в одній будівлі або в невеликому кампусі.
- Міська мережа (MAN): Охоплює велике місто або географічно обмежену територію, зазвичай між кількома локальними мережами.
- Глобальна мережа (WAN): Охоплює велику територію, зазвичай всю планету. Інтернет є прикладом глобальної мережі.

Фізична топологія:

- Зірка: Всі пристрої підключені до одного центрального комутатора або концентратора.
- Кільце: кожен комп'ютер з'єднаний з двома іншими, утворюючи фізичний кільцевий шлях.
- Шина: всі пристрої підключені до одного спільного каналу, по якому дані передаються на всі пристрої.

Архітектура:

- Клієнт-сервер: Комп'ютери поділяються на клієнти, які запитують ресурси, і сервери, які надають ці ресурси.

- Однорангова: Комп'ютери спілкуються один з одним без центрального сервера. Кожен комп'ютер може діяти як клієнт і сервер одночасно.

Призначення:

- Бізнес-мережі: Використовуються для підтримки бізнес-операцій, обміну даними та офісного спілкування.

- Домашні мережі: Встановлюються в домогосподарствах для підключення різних пристроїв, таких як комп'ютери, смартфони, телевізори тощо.

- Академічні мережі: Використовуються в академічних та наукових дослідженнях для обміну даними та співпраці між науковцями.

Технології передачі даних:

- Комутована мережа: Використовує комутатори для ефективної маршрутизації даних.

- Некомутована мережа: Використовує концентратори або хаби для передачі даних.

Програми:

- Інтернет: Глобальна мережа, яка об'єднує мільйони комп'ютерів і серверів для спільного використання даних і ресурсів.

- Інтранет: Внутрішня мережа, що використовується всередині організації для обміну інформацією та ресурсами.

- Екстранет: Мережа, яка з'єднує дві або більше організацій для спільного використання даних і ресурсів.

- Віртуальна приватна мережа (VPN): Забезпечує безпечний зв'язок через відкриті мережі, такі як Інтернет.

Розглянемо поняття інформаційної системи та її складові.

ІС - це складний комплекс апаратних і програмних засобів для збору, обробки, зберігання та передачі даних для вирішення широкого кола завдань. До основних компонентів сучасних інформаційних систем належать

- Дані: Інформація, яка збирається, зберігається і використовується в системі. Це може бути текст, числа, графіка, аудіо, відео та інші форми інформації.

-Програмне забезпечення: Додатки, програми та операційні системи, які використовуються для обробки та аналізу даних, виконання різних завдань і надання функціональності користувачам.
-Апаратне забезпечення: Фізичні пристрої та компоненти, такі як комп'ютери, сервери, датчики, мережеве обладнання та сховища, які використовуються для роботи інформаційної системи.
-Мережева інфраструктура: Інфраструктура для зв'язку та взаємодії між різними компонентами системи через мережу.
-Людські ресурси: Люди, які взаємодіють з інформаційною системою, включаючи адміністраторів, користувачів, розробників та технічну підтримку.
-Інструменти безпеки: Інструменти та методи захисту інформації та інфраструктури від несанкціонованого доступу, атак та загроз.
-Процеси: Інструкції та процедури, які визначають, як система повинна поводитися, наприклад, збирати, обробляти та поширювати дані, а також як користувачі повинні взаємодіяти з системою.
-Інструменти підтримки прийняття рішень: Інструменти для аналізу даних та надання інформації, необхідної для прийняття ефективних рішень.

Інформаційні системи можуть бути розроблені та налаштовані відповідно до конкретних потреб і вимог організацій та користувачів. Вони відіграють важливу роль в управлінні даними, автоматизації процесів і забезпеченні ефективності в різних сферах діяльності.

Сьогодні мережева активність користувачів Інтернету стає невичерпним джерелом цифрових даних, які є стратегічно важливими для широкого кола застосувань. Збільшення кількості цифрової інформації робить важливим аналіз цих даних з метою виявлення важливих залежностей, виявлення загроз та покращення процесів обслуговування користувачів. Розглянемо найпоширеніші технології мережевого моніторингу.

SNMP - це простий протокол управління мережею, який використовується для збору інформації про стан мережевих пристроїв, таких як маршрутизатори, комутатори, сервери тощо. SNMP - це технологія, призначена для забезпечення управління і

контролю пристроїв і додатків в мережі зв'язку шляхом обміну керуючою інформацією між агентами, розташованими на мережевих пристроях, і менеджерами, розташованими на станціях управління. SNMP визначає мережу як сукупність станцій мережевого управління і мережевих елементів (хостів, шлюзів і маршрутизаторів, термінальних серверів), які спільно забезпечують адміністративний зв'язок між станціями мережевого управління і мережевими агентами. Давайте коротко розглянемо архітектуру SNMP. Архітектура SNMP базується на клієнт-серверному підході і включає кілька ключових компонентів:
Станції управління (NMS - Network Management Station): Це комп'ютери або пристрої, які використовуються для моніторингу та управління мережею. Вони використовують програмне забезпечення SNMP для запиту і зчитування інформації від агентів.
Агенти: Агенти - це програмне забезпечення, яке встановлюється на мережеві пристрої, що підлягають моніторингу та управлінню. Агенти зберігають інформацію про стан пристрою і можуть виконувати дії за запитом NMS.
MIB (Management Information Base): MIB - це ієрархічна база даних, яка визначає, які параметри і дані можуть бути прочитані або встановлені на мережевих пристроях. Кожна MIB має унікальний ідентифікатор, відомий як OID (ідентифікатор об'єкта).
Протоколи SNMP: SNMP використовує різні версії протоколу, такі як SNMPv1, SNMPv2 і SNMPv3. Кожна версія включає різні функції та рівні безпеки. Наприклад, SNMPv3 підтримує автентифікацію та шифрування для захисту приватної інформації.
Транспортні протоколи: SNMP може використовувати різні транспортні протоколи для обміну даними між NMS та агентами. Найчастіше використовуються UDP (User Datagram Protocol) і TCP (Transmission Control Protocol).
Архітектура SNMP дозволяє мережевим адміністраторам віддалено контролювати стан пристроїв, налаштовувати їхні параметри та здійснювати дії з управління мережею. Це стандарт для управління мережею, який використовується в багатьох типах мереж і пристроїв.

NetFlow: NetFlow - це технологія, яка збирає статистику про трафік в мережі, включаючи інформацію про джерела, пункти призначення, протоколи та обсяги даних.

Записи потоку: NetFlow створює записи потоку, які містять інформацію про зв'язок між джерелом і пунктом призначення. Ця інформація може включати IP-адреси, порти, протоколи, обсяг трафіку і тривалість зв'язку.

Експортери потоку: Експортери потоку - це мережеві пристрої, які генерують і надсилають записи потоку до аналізатора NetFlow (зазвичай це система збору даних).

Збирачі потоку: Збирачі потоку - це сервери або пристрої, які отримують записи потоку від експортерів і зберігають дані для подальшого аналізу.

Аналізатори потоку: Аналізатори потоку - це програми або системи, які обробляють і аналізують дані NetFlow для відображення інформації про мережевий трафік, аналізу безпеки і виявлення аномалій.

Шаблони потоку: Шаблони потоку використовуються для визначення формату записів потоку та інформації, яка повинна бути включена в кожен запис.

Переваги технології NetFlow включають можливість моніторингу та аналізу мережевого трафіку, виявлення мережевих проблем, виявлення аномалій в мережевому трафіку і надання інформації для прийняття рішень щодо оптимізації мережі.

Дослідження пакетів: Сніфінг пакетів - це процес збору та аналізу мережевих пакетів, що проходять через мережевий інтерфейс. Це дозволяє вивчати трафік і виявляти проблеми в мережі. До основних компонентів цієї технології відносяться:

Мережева карта (NIC): Мережева карта - це мережева карта, яка дозволяє комп'ютеру або пристрою підключатися до мережі і перехоплювати мережеві пакети. Деякі мережеві карти можуть бути налаштовані в режимі "проміскуїтету", що дозволяє їм отримувати всі пакети, які проходять через мережевий сегмент, незалежно від їхнього призначення.

Програмне забезпечення для перехоплення пакетів: Це програмне забезпечення, яке працює з мережевою картою і надає можливість записувати, зберігати та аналізувати

перехоплені мережеві пакети. До популярних програм перехоплення пакетів належать Wireshark, tcpdump та інші.

Інструменти фільтрації та аналізу: Інструменти фільтрації та аналізу дозволяють користувачам вибирати пакети, які їх цікавлять, і аналізувати їхній вміст. Вони можуть застосовувати фільтри за протоколом, IP-адресою, портом та іншими параметрами.

Зберігання: Перехоплені пакети можна зберігати для подальшого аналізу. Зазвичай для цього потрібен значний обсяг дискового простору, оскільки мережевий трафік може бути великим і швидким.

Перехоплення пакетів можна використовувати для різних цілей, зокрема

Моніторинг мережі: Допомагає відстежувати і аналізувати мережевий трафік для виявлення мережевих проблем, включаючи перевантаження і аномалії.

Аналіз безпеки: Використовується для виявлення потенційних загроз і атак на мережу, таких як зломи і вторгнення.

Діагностика проблем: Допомагає виявити проблеми в мережі та швидко їх вирішити.

Збір та аналіз даних про використання мережі: Дозволяє вивчати, як користувачі використовують мережу, щоб приймати рішення щодо оптимізації ресурсів.

Моніторинг потоку: Моніторинг потоків - це метод, при якому мережевий трафік аналізується на основі потоків, а не окремих пакетів. Це дозволяє отримати загальну статистику про потік даних, що проходять через мережу. Розглянемо основні принципи цього методу:

Записи потоків: Моніторинг потоків передбачає створення записів потоків, які містять інформацію про кожен потік, включаючи джерело, призначення, протокол, порти, обсяг трафіку і тривалість зв'язку.

Експортери потоків: Експортери потоків - це мережеві пристрої, які створюють записи потоків і надсилають їх аналізаторам потоків або колекторам потоків.

Збирачі потоку: Збирачі потоку - це системи, які отримують дані від експортерів і зберігають їх для подальшого аналізу. Вони можуть агрегувати великі обсяги даних для створення загальних звітів і графіків.

Інструменти для аналізу потоків: Це програмне забезпечення, яке використовується

для обробки та аналізу записів потоку. Воно може включати інструменти для моніторингу безпеки, виявлення аномалій, визначення пропускної здатності тощо.
Моніторинг потоку дозволяє вирішувати різні завдання:
Моніторинг мережі: Допомагає відстежувати і аналізувати мережевий трафік, визначати пропускну здатність і споживані ресурси.
Аналіз безпеки: Використовується для виявлення потенційних загроз, зломів і аномалій в мережі.
Оптимізація ресурсів: Допомагає вирішувати проблеми зі споживанням ресурсів і приймати рішення щодо оптимізації мережі.
Планування мережі: Дозволяє аналізувати трафік для планування розширення мережі та вибору оптимальних ресурсів.
Моніторинг потоку - дуже корисний інструмент для мережевих адміністраторів для забезпечення ефективності та безпеки мережі.
Вибірка потоку: Вибірка потоку - це метод, при якому замість аналізу кожного пакета аналізується лише певна вибірка пакетів. Це зменшує кількість даних для аналізу, але при цьому надає загальну статистику.
Системи журналювання - це ключова технологія мережевого спостереження, яка передбачає збір, аналіз і зберігання журналів подій з різних джерел в інформаційних системах і мережах. Ця технологія важлива для моніторингу безпеки, діагностики помилок, виявлення аномалій та аудиту системи. Ось детальний опис систем логування:
Збір подій: Системи журналювання здатні збирати події з різних джерел, таких як операційні системи, програми, мережеве обладнання, брандмауери та інші. Сюди входять записи про вхідні та вихідні з'єднання, зміни конфігурації, доступ до файлів, аудит безпеки та багато інших подій.
Фільтрація та обробка: Після збору подій система реєстрації може фільтрувати і обробляти записи, усуваючи нецікаві або зайві події. Це допомагає зосередитися на важливих або підозрілих подіях.
Зберігання: Журнали подій зазвичай зберігаються в безпечних сховищах протягом певного періоду часу (зазвичай від кількох місяців до років). Це важливо для

довгострокового аналізу та аудиту.

Аналіз і пошук: Системи реєстрації подій надають засоби для аналізу та пошуку великого обсягу подій. Це дозволяє виявляти аномалії, атаки або незвичні шаблони в журналах подій.

Звітування та оповіщення: Системи реєстрації можуть створювати звіти на основі аналізу журналів і надсилати сповіщення адміністраторам або іншим особам, які приймають рішення, при виявленні підозрілих або критичних подій.

Відповідність та аудит: Системи реєстрації важливі для дотримання вимог безпеки та аудиту. Вони допомагають гарантувати, що система відповідає нормативним вимогам.

Системи журналювання використовуються в інформаційних системах і мережах для забезпечення безпеки, відновлення обслуговування, діагностики помилок і аудиту. Вони дозволяють адміністраторам та інженерам своєчасно виявляти і реагувати на події, що відбуваються в системі, тим самим підвищуючи безпеку і ефективність.

Системи виявлення та запобігання вторгнень (IDS/IPS): IDS і IPS використовуються для виявлення і блокування незвичайної або потенційно шкідливої активності в мережі.

IDS та IPS - це дві різні технології виявлення та управління вторгненнями в мережі та системи. Вони використовуються для забезпечення мережевої безпеки та обмеження незаконної або зловмисної діяльності в цифровому середовищі. Ось короткий опис кожної з цих технологій:

IDS: призначена для виявлення незаконної або аномальної активності в мережі або системі. Вона аналізує мережевий трафік, журнали подій або інші джерела для виявлення аномалій або атак. Коли IDS виявляє потенційне вторгнення або незвичайну активність, він генерує попередження або реєструє подію для подальшого аналізу. IDS не може активно реагувати на вторгнення або блокувати їх, а лише інформує адміністратора про їх виявлення.

IPS (Intrusion Prevention System - система запобігання вторгненням): IPS є розвитком IDS і забезпечує додатковий рівень захисту. Вона виявляє потенційні вторгнення або

аномальну активність так само, як і IDS, але на додаток до цього може активно реагувати і блокувати зловмисну активність. Це включає блокування IP-адрес, використання брандмауера для заборони доступу та інші заходи безпеки для запобігання атакам.

Основні відмінності між IDS та IPS:

Функція: IDS призначена для виявлення вторгнень і аномалій, в той час як IPS включає в себе функцію блокування і запобігання вторгненням.

Реагування: IDS надсилає сповіщення про виявлені атаки або аномалії, тоді як СПП може автоматично реагувати і блокувати зловмисну активність.

Активність: IDS - це пасивна система спостереження, в той час як IPS - це активна система запобігання.

Обидві технології важливі для забезпечення безпеки мережі та інформаційної системи. Вибір між IDS та IPS залежить від конкретних потреб безпеки та ризиків, які необхідно усунути. Багато організацій використовують обидві системи для комплексного захисту мережі.

2 АНАЛІЗ ІСНУЮЧИХ ТЕХНОЛОГІЙ, СПОСОБІВ ТА ЗАСОБІВ СПОСТЕРЕЖЕННЯ, АДМІНІСТРУВАННЯ І ОБРОБКИ ДАНИХ, ЩОДО МЕРЕЖЕВОЇ ПОВЕДІНКИ КОРИСТУВАЧІВ В РІЗНИХ СЕГМЕНТАХ СУЧАСНИХ ІС

2.1 Огляд існуючих технологій мережевого спостереження та узагальнення можливостей відомих способів і засобів перехоплення й екстракції даних

Пакетні аналізатори та мережеві інструменти:

1)Wireshark:

Плюси:

Інтерфейс користувача: Простий у використанні, зручний інтерфейс для аналізу пакетів.

Спільнота користувачів: Велика активна спільнота для підтримки та обміну досвідом.

Мінуси:

Великі обсяги даних: Вимагає значного обсягу дискового простору для зберігання даних.

Не завжди легко аналізувати: Може бути складним для аналізу для непосвячених через велику кількість інформації.

2) Проксі-сервери та перехоплення HTTP-трафіку:

Скрипаль:

Плюси:

Легко налаштувати: Простий у встановленні та використанні.

Інтерактивний аналіз: Можливість редагування HTTP-трафіку в режимі реального часу.

Мінуси:

Обмеження: Спеціалізується на HTTP, що обмежує висновки про загальний мережевий трафік.

Можливість обходу: Може бути вразливим до обхідних атак.

3) Системи виявлення вторгнень (IDS) і системи запобігання вторгнень (IPS):

Snort:

Плюси:

Велика спільнота: Активна спільнота користувачів і широкий спектр правил.

Робота в режимі реального часу: Здатність виявляти загрози в режимі реального часу.

Мінуси:

Хибні спрацьовування: Схильні до генерації хибнопозитивних тривог.

Складно налаштувати: Потребує досвіду для оптимізації та уникнення хибнопозитивних спрацьовувань.

4) Системи мережевого логування та аудиту:

Syslog:

Плюси:

Стандарт: Загальноприйнятий стандарт для передачі журналів подій у мережі.

Простота: Простий у використанні та розгортанні.

Мінуси:

Не в режимі реального часу: Не завжди надає інформацію в режимі реального часу.

Обмежені формати: Деякі реалізації можуть бути обмежені форматами журналів.

5) Аналіз трафіку та системи мережевої безпеки:

Cisco Stealthwatch:

Переваги:

Висока продуктивність: Здатність обробляти великі обсяги даних для виявлення загроз.

Комплексний огляд: Аналізує як стан мережі, так і активність користувачів.

Мінуси:

Вартість: деякі функції можуть бути дорогими для підприємств.

Складність: Вимагає досвіду для ефективного використання.

6) Системи для аналізу поведінки мережі:

Darktrace:

Плюси:

Використання ШІ: Використовує штучний інтелект для виявлення аномалій у поведінці.

Система в режимі реального часу: Надає можливість миттєво виявляти загрози.

Мінуси:

Хибні спрацьовування: Як і всі системи на основі штучного інтелекту, може генерувати помилкові спрацьовування.

Вартість: Впровадження та підтримка можуть бути дорогими.

7) Системи моніторингу корпоративної мережі:

SolarWinds Network Performance Monitor:

Плюси:

Автоматизація: Допомагає автоматизувати процеси моніторингу та реагування.

Сповіщення: Надає можливість отримувати сповіщення про проблеми в мережі.

Мінуси:

Складність налаштування: Вимагає часу та досвіду для повного налаштування.

Вартість: Повна функціональність може бути дороговартісною для малого бізнесу.

2.2 Визначення і обґрунтування переліку персоніфікованих параметрів мережевої активності для різних категорій користувачів (веб-додатків)

Для «демон-юнітів», які використовуються для накопичення та узагальнення даних про мережеву активність користувачів і синтезу їхніх поведінкових профілів, телеметричні, поведінкові та пошуково-рефлекторні параметри мають вирішальне значення. Розглянемо докладніше ці параметри та їхню важливість:

Телеметричні параметри:

Приклад:

Телеметричні дані можуть включати інформацію про те, як довго використовуються додатки, як часто взаємодіють з певними функціями, скільки даних передається тощо.

Переваги:

Моніторинг продуктивності:

Дозволяє виміряти ефективність роботи «демон-юніта» і виявити можливі проблеми з продуктивністю.

Аналіз користувацького досвіду:

Дозволяє вирішити проблеми з інтерфейсом користувача та покращити загальний користувацький досвід.

Недоліки:

Конфіденційність користувачів:

Збір телеметричних даних може порушити конфіденційність користувача, тому важливо забезпечити анонімність і безпеку даних.

Поведінкові параметри:

Приклад:

Поведінкові дані можуть включати інформацію про типові дії користувача, взаємодію з різними функціями «демон-юніту», історію введення та реакції на певні стимули.

Переваги:

Виявлення аномалій:

Дозволяє виявляти незвичні або підозрілі моделі поведінки, які можуть свідчити про потенційні загрози.

Персоналізація:

Надає можливість адаптувати «демон-юніта» до унікальних потреб користувачів на основі їхнього індивідуального стилю використання.

Мінуси:

Хибні спрацювання:

Система може небажано реагувати на звичайні зміни в поведінці користувача, що може призвести до хибних спрацювань.

Параметри пошукового рефлексу:

Приклад:

Це можуть бути дані про те, як користувачі взаємодіють з функціями пошуку, які запити вони вводять і як система реагує на ці запити.

Плюси:

Оптимізація пошуку:

Дозволяє підвищити ефективність і релевантність результатів пошуку на основі звичок користувачів.

Аналіз попиту:

Дає можливість зрозуміти, які інформаційні запити користуються найбільшою популярністю серед користувачів.

Мінуси:

Ризик недостатнього рівня конфіденційності:

Збір пошукових запитів може становити ризик порушення приватності користувачів, тому важливо бути обережним при обробці цих даних.

Підсумовуючи, можна сказати, що телеметрія, поведінкові та пошукові параметри важливі для «демон-юнітів» для оптимізації продуктивності, покращення користувацького досвіду та забезпечення персоналізації. Однак збором і обробкою цих даних потрібно ретельно керувати, беручи до уваги конфіденційність користувачів і мінімізуючи можливі ризики хибних спрацьовувань або недостатнього захисту даних.

2.3 Дослідження процедур накопичення і узагальнення даних мережевої активності користувачів для синтезу їх поведінкових профілів – «демон-юнітів».

Поведінкові профілі використовуються для успішного відстеження загроз і запобігання спробам несанкціонованого доступу. Поведінковий профіль користувача в Інтернеті - це детальний образ користувача, який відображає його звички, інтереси, взаємодії та інші характеристики, що виникають під час його діяльності. Ці профілі створюються шляхом збору та аналізу різних даних, які можуть включати відвідані веб-сайти, використані додатки, поведінкові патерни тощо.

Збір даних для створення поведінкових профілів:

Файли cookie: Файли, які зберігаються на комп'ютері користувача і містять інформацію про його взаємодію з веб-сайтами.

IP-адреси: Унікальні ідентифікатори пристрою, що використовуються для підключення до Інтернету.

Ідентифікатори пристроїв: Унікальні коди, що використовуються для ідентифікації конкретного пристрою.

Дані аналітичного відстеження: Інформація про взаємодію користувача з веб-сайтами, додатками та іншими цифровими сервісами.

Соціальні мережі: Дані, які користувач надає через свої акаунти в соціальних мережах.

Пошукові запити: Інформація про те, що користувач шукає в Інтернеті.

Основні параметри профілювання:

Демографічна інформація: Вік, стать, місце проживання.

Інтереси та хобі: Інформація про те, чим цікавиться користувач.

Поведінкові патерни: Як користувач взаємодіє з онлайн-контентом і сервісами.

Покупки та транзакції: Інформація про покупки та фінансові операції.

Лояльність до бренду: Реакція користувача на певний бренд або продукт.

Як такі профілі допомагають уникнути кібератак:

Виявлення аномалій: Аналіз поведінкових профілів може допомогти виявити аномалії та незвичайну активність, які можуть бути ознаками кібератак.

Аутентифікація: Поведінкові характеристики можуть бути використані для аутентифікації користувача, що додатково підтверджує його особу.

Персоналізована безпека: На основі профілів можна розробляти персоналізовані стратегії безпеки і попереджати користувачів про можливі ризики.

Моніторинг загроз: Аналіз поведінкових даних дозволяє швидко реагувати на нові загрози та вдосконалювати системи безпеки.

Залежно від типу бізнесу або організації до профілю додаються різні параметри.

Корпоративна безпека: В контексті корпоративної інформаційної безпеки поведінкові профілі можуть включати моніторинг активності користувачів в корпоративних мережах. Наприклад, система може аналізувати типові години доступу користувачів, звичайні місця доступу та частоту зміни паролів. Виявлення аномальних змін у цих шаблонах може сигналізувати про можливий несанкціонований доступ або кібератаку.

Моніторинг доступу до критично важливих даних: У випадках, коли компанія обробляє конфіденційну інформацію або великі обсяги даних, створення поведінкових профілів для кожного користувача стає ключовим. Наприклад, система може фіксувати, які файли переглядає користувач, які дані він змінює і як часто взаємодіє з критичними ресурсами. Це дозволяє швидко виявити надмірні права доступу або незвичайну активність, яка може бути підозрілою.

Поведінкові фактори ризику в кібербезпеці: У кібербезпеці профілювання може враховувати поведінкові фактори ризику. Наприклад, система може аналізувати, як часто користувачі відкривають фішингові електронні листи або переходять за посиланнями в них. Це допомагає виявити користувачів, які можуть бути більш вразливими до атак соціальної інженерії.

Управління привілеями: Системи також можуть використовувати поведінкові профілі для управління привілеями користувачів. Наприклад, якщо користувач демонструє низький рівень активності в системі протягом тривалого часу, його рівень доступу може бути автоматично обмежений, щоб підвищити загальний рівень безпеки.

Прогнозування загроз: Використання поведінкової аналітики може допомогти в прогнозуванні потенційних загроз для компаній. Наприклад, аномальне зростання активності користувачів, особливо навколо критично важливих систем, може свідчити про несанкціонований доступ або спроби злому.

Ці конкретні застосування поведінкових профілів в інформаційних системах компаній допомагають підвищити ефективність кібербезпеки, уникати загроз і реагувати на події в режимі реального часу.

У контексті кібербезпеки створення поведінкового профілю базується на аналізі різних параметрів, які можуть вказувати на аномалії або підозрілу активність. Нижче наведені деякі з ключових параметрів, які зазвичай розглядаються:

Типова поведінка: Нормальна діяльність. Аналіз того, що є "нормальним" для конкретного користувача або системи.

Стандартні години доступу: Визначення звичайного робочого графіка користувача.

Взаємодія з ресурсами: Частота і типова активність: Як часто користувач взаємодіє з різними ресурсами та сервісами.

Зміни у файловій системі: Виявлення незвичайних змін або видалених файлів.

Системні дії та активність: Процеси та служби: Відстежуйте активність процесів і служб у системі.

Події автентифікації: Реєструє входи і виходи з системи, зміни паролів.

Мережева активність: Підключення та взаємодія з іншими пристроями: Аналіз мережевої активності користувача.

Несподівані з'єднання: Виявлення незвичайних мережевих підключень.

Діяльність у сфері безпеки: Запуск антивірусних та інших програм безпеки. Перевірка стану антивірусного програмного забезпечення.

Потреба в підвищенні прав: Визначає, коли користувач намагається отримати додаткові привілеї.

Спроби несанкціонованого доступу: Неправильне введення пароля: Моніторинг повторного неправильного введення пароля.

Несподівані входи: Виявлення входів з несподіваних місць або пристроїв.

Діяльність у закритих ресурсах: Доступ до конфіденційної інформації: Виявлення спроб доступу до конфіденційних даних.

Реагування на загрози: Час реагування на оповіщення: Аналіз швидкості реагування на тривоги та інциденти безпеки.

Створення поведінкових профілів передбачає аналіз цих параметрів і визначення того, що є "нормальним" для конкретного користувача або системи. Аномалії можуть вказувати на можливі кіберзагрози або вразливості, які потребують уваги та відповідних заходів безпеки.

У контексті операційних систем і взаємодіючи з поведінковими профілями, демони можуть збирати інформацію з різних джерел і компонентів системи. Важливо зазначити, що конкретні методи можуть залежати від конкретного використання, типу демона, а також вимог до приватності та конфіденційності. Нижче наведено кілька поширених методів збору інформації:

Системні журнали: Демони можуть записувати події та дії у системні журнали.

Інформація про запуск і завершення роботи:Дані про те, коли було запущено або завершено роботу демона.

Активність у часовому шарі: Інформація про активність демона у різні періоди часу.

Мережева активність: Модуль демона може відстежувати мережеву активність, визначаючи, які ресурси використовує користувач і з якими серверами він взаємодіє.

Аналіз використання ресурсів: Вимірювання використання ресурсів: Моніторинг використання процесора, пам'яті та інших системних ресурсів користувачем та іншими демонами.

Взаємодія з файловою системою: Запис інформації про файли: Інформація про те, які файли користувач відкриває, змінює або створює.

Відстеження змін у каталогах: Відстежуйте зміни у важливих каталогах, які можуть свідчити про активність.

Взаємодія з системою безпеки:

Журнали безпеки та інцидентів: Аналіз журналів для виявлення потенційних загроз або спроб несанкціонованого доступу.

Ключові події та вхідні дані: Збір інформації про ключові події, такі як команди, введені користувачем, або реакції на певні події.

Ці методи допомагають «демон-юнітам» збирати інформацію про те, як користувачі або інші компоненти системи взаємодіють з операційною системою. Використовуючи ці дані, можна створювати поведінкові профілі, які дозволяють аналізувати і реагувати на зміни в системі.

У світі кібербезпеки «демон-юніти» виступаючи фоновими компонентами операційних систем, є невід'ємною частиною моніторингу системи та збору інформації. Вони відповідають за взаємодію з системним середовищем, обробку подій і збір важливих даних, які можуть бути використані для створення поведінкових профілів.

Деякі з ключових джерел інформації, які використовують «демон-юніти» , включають системні журнали, які фіксують активність системи та її зміни, а також мережеву активність, що дозволяє їм аналізувати взаємодію зовнішніх ресурсів.

Крім того, вони досліджують використання таких ресурсів, як процесор і пам'ять, а також взаємодію з файловою системою, включаючи зміни і маніпуляції з файлами і каталогами.

Інформація, отримана від «демон-юніта» є основою для створення поведінкових профілів, які визначають типові патерни активності користувача або системних процесів. Ці профілі використовуються для виявлення аномалій, ідентифікації потенційно небезпечних ситуацій та взаємодії з кіберзагрозами. Важливим викликом у використанні даних, зібраних «демон-юнітами» , є необхідність дотримання етичних норм і законів щодо конфіденційності та приватності. Врахування цих аспектів є обов'язковим для забезпечення захисту особистої інформації користувачів.

Таким чином, «демон-юніти» відіграють ключову роль у виявленні, аналізі та реагуванні на загрози в інформаційному середовищі, забезпечуючи підвищений рівень кібербезпеки та ефективне управління системними ресурсами.

3 СТВОРЕННЯ ТА ДОСЛІДЖЕННЯ ТЕСТОВОГО ПРОГРАМНОГО РІШЕННЯ ДЛЯ ПЕРЕХВАТУ, ЕКСТРАКЦІЇ І УЗАГАЛЬНЕННЯ ПОВЕДІНКОВИХ ДАНИХ ПРО МЕРЕЖЕВУ АКТИВНІСТЬ КОРИСТУВАЧІВ

3.1 Синтез структури алгоритму екстрактора даних поведінкового профілю для одиночних мережевих користувачів

Для вирішення зазначених вище завдань необхідно визначити структуру та провести аналіз функції основних елементів алгоритму автоматизованої компіляції поведінкових профілів мережевих користувачів. Стисло розглянемо основні елементи прототипу алгоритму автоматизованого синтезу одиночного «демон-юніту».

Модуль «параметричного програматору»: - забезпечує регламентацію бажаних (спостережуваних) властивостей мережевої поведінки для визначених категорій об'єктів (користувачів та/або вузлів). Перед початком процесу компіляції важливо визначити перелік та зміст бажаних властивостей поведінки для створюваного «демон-юніту». Перш за все, це стосується типу і обсягів даних, які потрібно зібрати, тривалості та/або періодів сеансів спостереження, «точок» збору інформації, а також методи аналізу й візуалізації даних.

Модуль «екстракції даних»: - відповідає за вилучення та збір необхідних даних від цільових користувачів мережі. Цей модуль необхідний для збору даних у режимі реального часу, та має бути налаштований для накопичення визначених раніше типів інформації на основі спостережуваних (або бажаних) поведінкових властивостей мережевих користувачів. До потенційних проблем при розробці цього модулю, можна віднести труднощі зі збором даних із різних типів мереж і систем (пристроїв) . Враховуючи різноманіття потенційних кінцевих реалізацій, цей модуль може знадобитися оновити та/або змінити для підтримки нових типів мережевих архітектур або операційних систем. Крім того, можуть

виникнути проблеми з обсягом зібраних даних, що може призвести до проблем із продуктивністю або надмірної надлишковості вихідних даних, цього модулю, що потребує відповідних коригувань параметрів його роботи.

Модуль аналізу даних: - забезпечує аналіз накопичених даних та виявлення потрібних закономірностей і фактичних тенденцій процесів, що спостерігаються. Цей модуль повинен обробляти автономні кластери даних (що персоніфіковані за типом/категорією користувачів), генерувати звіти та забезпечувати зручну вибіркову візуалізацію подій, для контролю основних параметрів поведінкового профілю цільових користувачів. Модуль аналізу даних «може зіткнутися» з проблемами під час визначення закономірностей і тенденцій через слабку формалізацію та/або невизначеність певних критеріїв, щодо зібраних даних. Також, можуть виникнути певні труднощі з точністю вихідних даних, що може призвести до неточних прогнозних оцінок спостережуваних процесів. Крім того, може знадобитися оновити або змінити діючи налаштування цього модулю для підтримки нових типів даних для кожного з об'єктів (ресурсів) спостереження.

Модуль поведінкового моделювання: - відповідає за формування основних параметрів роботи для створюваного «демон-юніту» цільового користувача (або ресурсу) на основі узагальнення раніше отриманої інформації. Цей модуль повинен вилучати потрібні дані з модулю «аналізу даних» та синтезувати відповідні програмні уставки для відтворення потрібного сценарного полю цільової категорії «демон-юнітів» . Є найбільш складним у виконанні модулем, з одного боку через важливість коректного відтворення поведінки, а з другого через необхідність розробки графічної оболонки для контролю параметрів автоматизованої компіляції виконавчої частини одиночних «демон-юнітів» . Цей модуль може зіткнутися з труднощами точності імітування поведінки користувача, саме через складність формалізації можливих станів людської поведінки. Гості, адміністратори мережі, віддалені користувачі, облікові записи служб тощо, все це потенційно різні моделі поведінки, які потрібно структурувати. В цьому сенсі, можуть виникнути певні проблеми з точністю прогнозів стосовно змісту параметричної структури персоніфікованих даних користувачів, що може

призвести до хибних (атипових/нешаблонних) дій для створюваного «демон-юніту»

Таким чином, цей модуль має дозволити адміністратору процесу контролювати бажані властивості поведінки та, відповідно, корегувати модулі «параметричного програматору» і «екстракції даних». Модуль може знадобитися оновити для підтримки нових поведінкових властивостей та/або більш ретельних (таргетованих) налаштувань профілю «демон-юніту» з врахуванням нових відомостей, стосовно останніх загроз безпеки.

Інтеграційний модуль: - після того, як всі інші модулі розроблені, їх потрібно інтегрувати в єдиний комплекс. Ця інтеграція повинна бути здійснена таким чином, щоб гарантувати безперервну роботу алгоритму автоматизованої компіляції поведінкового профілю – «демон-юніту» та безперебійний зв’язок (трафік даних) окремих модулів між собою. Інтеграційний модуль може зіткнутися з труднощами безперебійної взаємодії різних модулів.

Модуль захисту даних: - модуль забезпечує захист даних, які отримані у модулі «екстракції та збору» даних (фактично, сховища оригіналів лог-файлів поведінкової телеметрії). Він повинен унеможливлювати потенційні спроби витоку зібраної інформації. Якщо ж він не виконав своїх функцій, то цій модуль повинен повідомити про це адміністратора процесу, та «запропонувати» кроки, щодо вирішення ситуації. Цей модуль слід вважати одним з основних, так як акумулює в собі чутливі (персоніфіковані) відомості, щодо мережної активності реальних груп мережевих користувачів (ресурсів).

Модуль конфігурації: - модуль конфігурації забезпечує налаштування та зберігання користувальницького профілю адміністратора процесу, та забезпечення потрібних характеристик графічного інтерфейсу та мережевої взаємодії в разі використання віддаленої консолі управління.

Спрощена структура прототипу алгоритму автоматизованого синтезу одиночного «демон-юніту», представлена.

Рисунок 3.1 - Спрощена структурна схема алгоритму

етап №1 − відбувається використання модулів конфігурації та параметричного програматору, та здійснюється налаштування усього нашого алгоритму.

етап №2 − використовуються модулі екстракції та захисту даних, відбувається накопичення та захист інформації про користувача.

етап №3 − використовується модуль аналізу даних. Відбувається сепарація (фільтрація) потрібних відомостей та спрощення отриманих даних.

етап №4 − використовується модуль поведінкового моделювання. Це найскладніший етап в якому відбувається створення аватару користувача з отриманими (накопиченими) раніше параметрами.

етап №5 − використовується модуль інтеграції. На цьому етапі відбувається об'єднання усіх отриманих раніше даних у єдину програму (профіль «демон-юніту»).

Через великий обсяг аналізованої інформації, вкрай важливо коректно визначити перелік параметрів, які будуть вилучатися у модулі екстракції даних Попередньо слід виділити 2 групи параметрів - типові, та не типові для даної області використання.

Типові параметрі − параметри, які широко використовуються у інших подібних юнітах/програмах/процесах. До них слід віднести:

- параметри геолокації пристрою;
- параметри служби DNS;
- ідентифікатори (SSID) характерних точок доступу Wi-Fi;
- кількість змін IP на добу;
- хронометраж обсягів вхідного та вихідного трафіку для всіх видів мережевих додатків;
- середній обсяг файлів, що зберігаються;
- параметри використання хмарної пам'яті;
- частота ініціації мережевих комунікацій;
- тижневий та добовий розподіл часу мережевої роботи по типам онлайн сервісів та служб;
- використовуваний (-ні) браузер;
- історія отримання мережевих ідентифікаторів сеансу, як докази проведених онлайн сесій;
- характерні склад та послідовність пошукових запитів (мережева історія);
- час перебування на характерних Інтернет ресурсах.

Нетипові параметрі - параметри, які взагалі не використовуються при зборі інформації іншими «демон-юнітами» або використовуються лише частково.

- тип текстової розкладки з хронологією перемикання;
- роздільна здатність екрану;
- параметри чутливості миші та/або сенсорного екрану;
- параметри служби часу з прив'язкою до часового поясу;
- середній темп набору символів;
- середня пауза між введенням окремих символів при використанні режиму вводу символів тощо.

Слід зауважити, що в модулі аналізу даних нетипові параметри матимуть вищий пріоритет, через важливість отримання саме користувальницьких (таргетованих) налаштувань системи.

3.2 Розробка та дослідження тестової програмної моделі екстрактора даних в версії одиночного «демон-юніту»

Постановка задачі - потрібно розробити програмне рішення для вилучення даних користувача з системи Windows. Дані, які нам потрібні, можна розділити на 3 основні категорії:

1) Телеметрія - це вилучення/перехоплення, накопичення та узагальнення фактичних характеристик мережевої активності реального користувача в розрізі технічних параметрів, що визначають історію мережевих сесій.

2) Поведінка - це вилучення/перехоплення, накопичення та узагальнення фактичних характеристик мережевої активності реального користувача в частині обробки антропогенних властивостей, що визначають історію (мережевий слід) сеансів.

3) Пошукове відображення - вилучення/перехоплення, накопичення та узагальнення фактичних характеристик мережевого серфінгу реального користувача, які визначають історію (мережевий слід) його мережевих сесій.

Алгоритм вилучення даних буде використовувати інформацію про параметри телеметрії користувача в ОС Windows для отримання інформації про параметри телеметрії користувача:

Журнали подій:

Журнали подій містять інформацію про різні аспекти роботи системи. Вони можуть містити дані про входи і виходи користувачів, використання ресурсів, події входу в систему, встановлення програм тощо.

Журнали аудиту безпеки:

Журнали аудиту безпеки містять інформацію про спроби входу в систему, зміни прав доступу, спроби видалення файлів тощо. Вони дозволяють відстежувати дії користувачів з великими обсягами телеметричних даних.

Лічильники продуктивності Windows:

Windows має набір лічильників продуктивності, які вимірюють різні параметри системи, такі як використання процесора, пам'яті, мережевого трафіку тощо. Ці дані можуть слугувати параметрами телеметрії.

Інструменти керування Windows (Windows Management Instrumentation, WMI):

WMI надає інтерфейс для збору різноманітної системної інформації, зокрема інформації про процеси, служби, мережеві з'єднання та інші системні параметри.

Аудит файлів і папок:

Увімкнення аудиту для певних папок і файлів дозволяє відстежувати спроби доступу та зміни до цих об'єктів, що може бути корисним для виявлення активності користувачів.

API для збору даних:

Використовуйте спеціалізовані API для отримання телеметричних даних, наприклад, API для служб безпеки, параметрів мережі тощо.

Аналіз мережевого трафіку:

Моніторинг та аналіз мережевого трафіку може надати інформацію про взаємодію системи з мережею, таку як мережеві сеанси, використання протоколів тощо.

Для отримання поведінкових параметрів буде використано наступне:

Моніторинг додатків:

Використовуйте інструменти моніторингу для відстеження активності конкретних програм, використання ресурсів і змін у конфігурації.

Реєстр Windows:

Зміни в реєстрі можуть вказувати на втручання користувача в налаштування системи та програм.

Моніторинг мережевого трафіку:

Аналіз мережевого трафіку може надати інформацію про взаємодію користувача з Інтернетом і мережевими ресурсами.

Кейлоггінг:

Там, де це дозволено і не суперечить закону, клавіатурні шпигунські програми можуть використовуватися для відстеження тексту, що вводиться користувачем.

Програмне забезпечення для моніторингу активності користувача:

Для відображення пошуку - це:

Аналіз історії браузера:

Вивчення історії браузера для виявлення пошукових запитів, введених користувачем, і відвіданих веб-сайтів.

Аналіз кешу браузера:

Використання інформації з кешу браузера для отримання відомостей про збережені сторінки та ресурси, які можуть вказувати на шаблони пошукової активності.

Моніторинг DNS-запитів:

Аналіз DNS-запитів для визначення відвіданих веб-сайтів і доменів, які пов'язані з пошуковими запитами.

Використання API пошукових систем:

Взаємодія з API пошукових систем для отримання детальної інформації про пошукові запити та рекомендації.

Аналіз системних журналів:

Вивчення системних журналів для виявлення запитів і активності, пов'язаної з пошуковими операціями.

Використання проксі-серверів:

Моніторинг трафіку через проксі-сервери для аналізу взаємодії користувачів з веб-ресурсами та пошуковими системами.

Аналіз файлів cookie:

Вивчення інформації з файлів cookie, які можуть містити дані про пошукові запити та інші взаємодії користувачів з веб-сайтами.

Використання спеціалізованих програм для аналізу трафіку:

Використання програм та інструментів для аналізу мережевого трафіку з метою виявлення пошукової активності користувачів.

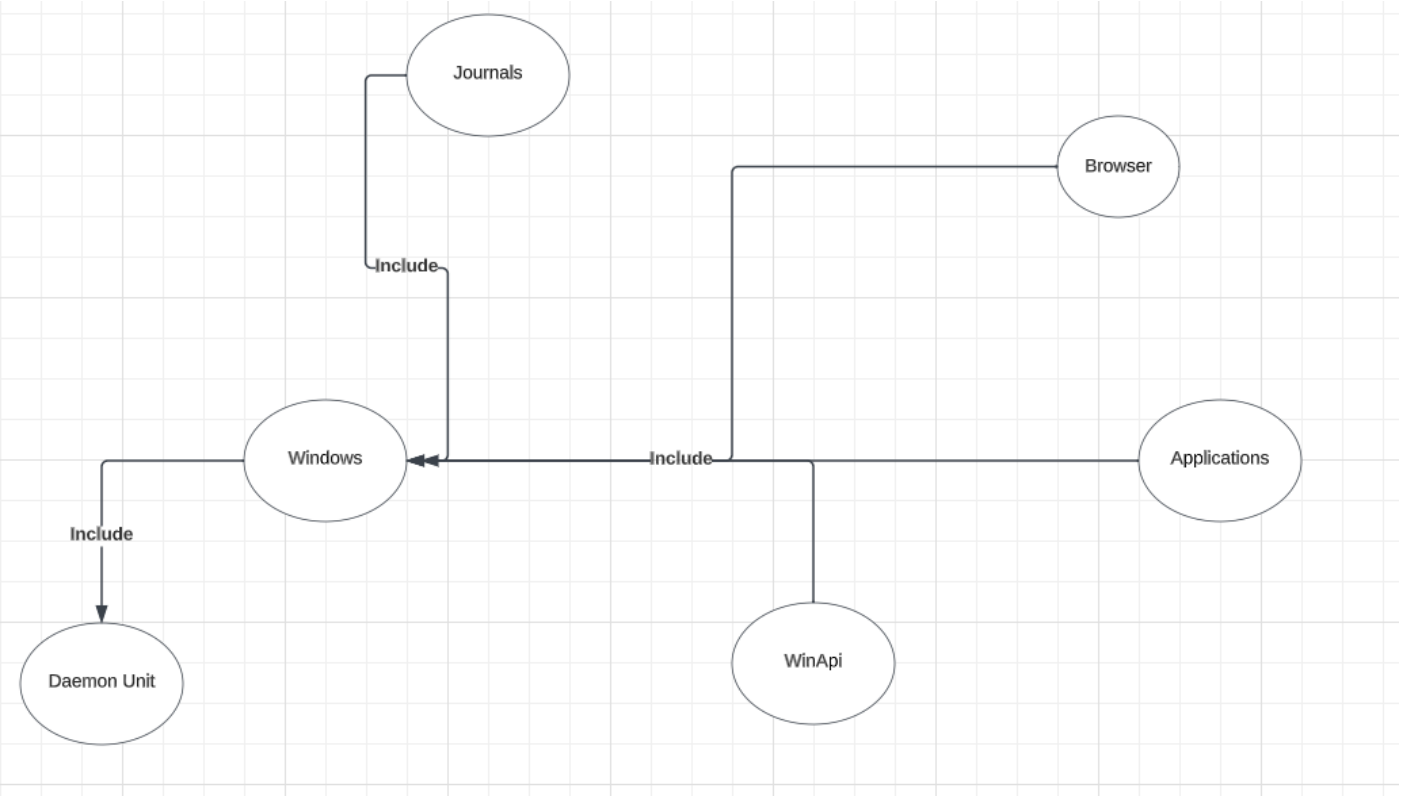

Рисунок 3.2 Діаграма обробки інформації

Info

Файл Изменить Просмотр

```
1.No port found 2.192.168.1.100 3.1920x1200 4.Wed Jul 19 23:42:34 2023 5.ru-RU 6.480 7.48.46739, 35.01464 8.Spotify 9.0 10.2.4MB
1.No port found 2.192.168.1.100 3.1920x1200 4.Wed Jul 19 23:42:44 2023 5.en-EN 6.320 7.48.46739, 35.01464 8.Telegram 9.0 10.3.5MB
1.No port found 2.192.168.1.100 3.1920x1200 4.Wed Jul 19 23:42:53 2023 5.uk-UK 6.480 7.48.46739, 35.01464 8.Notes 9.0 10.4.6MB
1.No port found 2.192.168.1.100 3.1920x1200 4.Wed Jul 19 23:42:53 2023 5.ru-RU 6.1200 7.48.46739, 35.01464 8.Spotify 9.0 10.5.7MB
1.No port found 2.192.168.1.100 3.1920x1200 4.Wed Jul 19 23:42:54 2023 5.ru-RU 6.480 7.48.46739, 35.01464 8.Opera 9.0 10.4.2MB
```

Рисунок 3.3 - Датасет з інформацією

На рис. 3.3 наведено приклад результатів запуску алгоритму збору інформації 5 разів. Стовпчик 1 відповідає за геолокацію за допомогою мережі. Стовпчик 2 відповідає за отримання IP. Стовпчик 3 відповідає за роздільну здатність екрану. 4 показує, в який час був запущений алгоритм збору інформації. 5 стовпчик відповідає за розкладку клавіатури. 6 стовпчик відповідає за DPI миші. 7 стовпчик відповідає за отримання інформації з реєстру Windows про геолокацію пристрою (інформація вноситься до реєстру Windows, коли будь-яка програма запитує дозвіл на геолокацію). 8 стовпчик показує останню програму. Колонка 9

показує розмір збережених файлів за період часу між останнім запуском алгоритму збору інформації.

3.3 Формування пропозицій щодо можливостей компіляції «демон-юнітів» з заданим переліком їх властивостей

1) «Демон-юніт» не повинен завантажувати систему. Для цього краще вибрати мову програмування, яка перетворює код безпосередньо в машинну мову.

2) Замість використання готових бібліотек краще написати власні функції, які будуть працювати з Windows API, це прискорить виконання програми і зробить інтерфейс більш зрозумілим.

3) Модульність - блок «демон-юніту» повинен складатися з незалежних частин, які можна легко замінити/вдосконалити без шкоди для основних функцій. Також алгоритм повинен легко інтегруватися в готові системи.

4) При компіляції «демон-юнітів» із заданим переліком їх властивостей, він не повинен мати інтерфейсу. Користувач може тільки погіршити ситуацію, не надавши можливості створити повний поведінковий профіль.

5) Набори даних, де зберігається будь-яка інформація про користувача, повинні бути зашифровані. Алгоритм шифрування повинен бути стійким. Найкраще підійде симетричний шифр, оскільки вам доведеться часто звертатися до набору даних і оновлювати дані.

4 СИНТЕЗ ПРОТОТИПУ АЛГОРИТМУ ПРОГРАМНОГО ІМІТАТОРУ ДЛЯ КОМПІЛЯЦІЇ ОДИНИЧНОГО «ДЕМОН-ЮНІТУ» МЕРЕЖЕВОГО КОРИСТУВАЧА З ЗАДАНИМИ ПОВЕДІНКОВИМИ ВЛАСТИВОСТЯМИ

4.1 Синтез структури алгоритму автоматизованої компіляції поведінкового профілю дослідного прототипу мережевого користувача

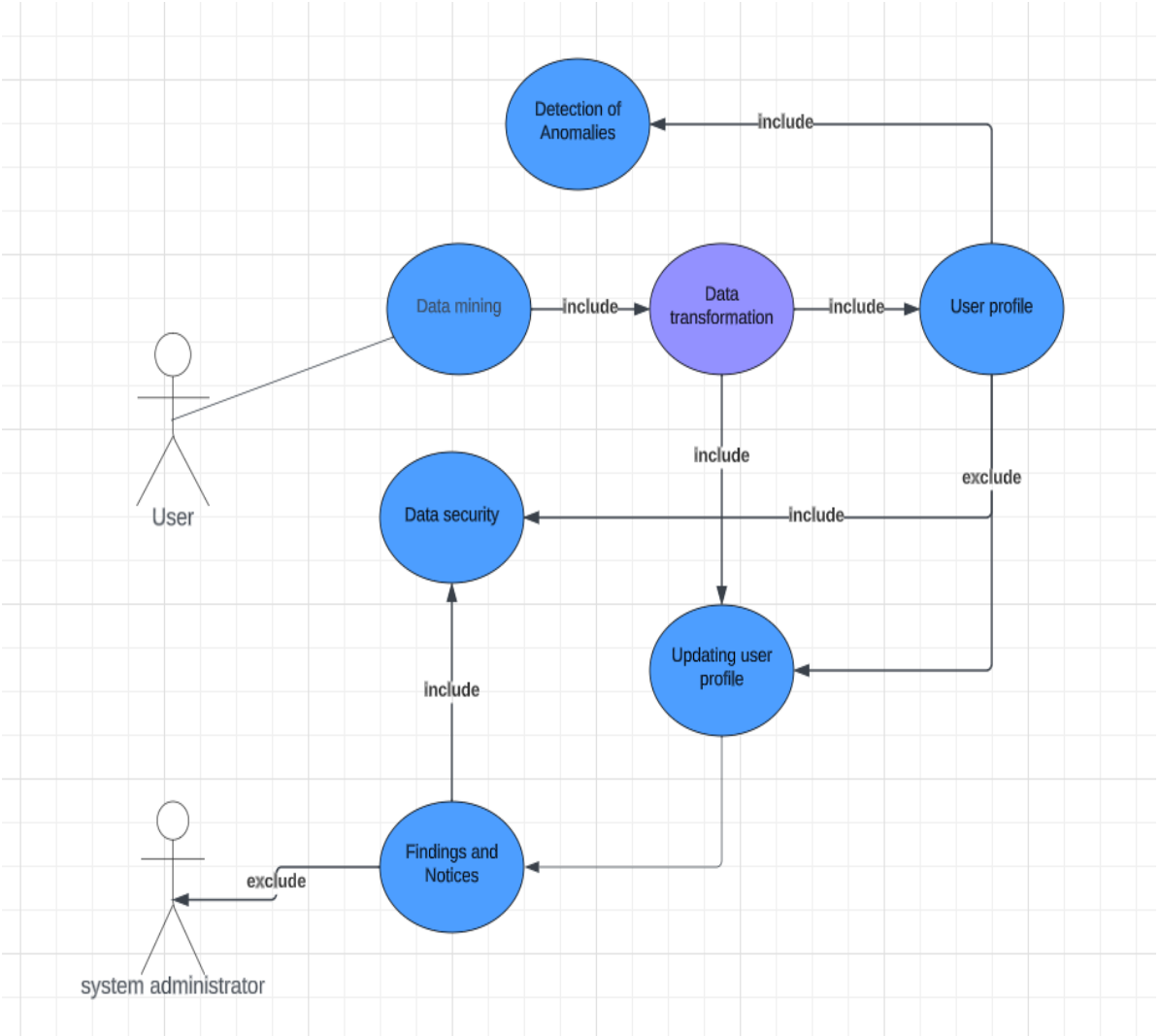

Рисунок 4.1 - Use case діаграма алгоритму

Алгоритм включає наступні кроки

1) Збір даних - здійснюється на основі алгоритму, описаного в пункті 3.

2) Виявлення аномалій - етап виявлення аномалій у поведінковому профілі користувача є ключовим компонентом системи безпеки, спрямованим на своєчасне

виявлення потенційно небезпечних або неочікуваних змін в активності користувача. Розглянемо докладніше кожен підетап цього етапу:

- Визначення норм зони:

Перед початком виявлення аномалій система визначає норми зон для різних аспектів активності користувачів. Це може включати типові години входу в систему, типові програми, що використовуються, та інші характеристики.

- Моніторинг активності:

Система безперервно відстежує активність користувача на основі побудованого поведінкового профілю. Це включає аналіз типових дій та взаємодій.

- Створення базового шаблону:

На початку процесу виявлення аномалії система генерує базовий шаблон поведінкового профілю, який вважається стандартним. Цей шаблон використовується для порівняння з поточною активністю.

- Порівняння з базовим шаблоном:

Під час моніторингу система постійно порівнює поточну активність з базовим шаблоном. Якщо спостерігаються значні відхилення, це може свідчити про аномалію.

- Використання методів машинного навчання:

Методи машинного навчання використовуються для підвищення точності виявлення аномалій. Алгоритми класифікації та кластеризації можуть виявити складні взаємозв'язки в активності користувачів.

- Ідентифікація сценаріїв ризику:

Система ідентифікує сценарії ризику, пов'язані з виявленими аномаліями. Це можуть бути невідомі або підозрілі програми, надмірна взаємодія з системними ресурсами та інші фактори.

- Генерація оповіщень та заходів безпеки:

При виявленні аномалій система генерує оповіщення для адміністраторів безпеки. При необхідності вживаються заходи безпеки, такі як автоматичне блокування підозрілих дій або примусовий вихід з системи.

- Аналіз та оптимізація помилкових спрацьовувань:

Система аналізує випадки, коли аномалії виявляються, але виявляються необґрунтованими (помилкові спрацьовування). На основі цього аналізу система може оптимізувати параметри виявлення аномалій, щоб зменшити кількість хибних спрацьовувань.

Цей процес виявлення аномалій у поведінковому профілі допомагає ефективно виявляти потенційні загрози безпеці та надмірно ризиковані дії користувачів.

3) Оновлення моделі - Етап оновлення моделі поведінкового профілю без використання машинного навчання передбачає процес адаптації до нових даних та умов активності користувача. Розглянемо цей процес детальніше:

- Збір та обробка нових даних:

Система постійно збирає нові дані про активність користувача. Це може бути інформація про використовувані додатки, часові з'єднання, параметри телеметрії та інші важливі характеристики.

- Аналіз та виявлення нових закономірностей:

Система аналізує нові дані, щоб виявити нові шаблони та особливості в діяльності користувача. Важливими аспектами можуть бути нові програми, частота використання або зміни в робочому графіку.

- Адаптація зональних норм:

На основі нових даних система адаптує зональні норми, які визначають типові характеристики активності користувачів. Наприклад, зміни в робочому часі або типових програмах можуть спричинити адаптацію цих норм.

- Оновлення поведінкового профілю:

На основі аналізу нових даних і адаптації норм система оновлює поведінковий профіль користувача. Це може включати зміни в параметрах, що описують активність користувача.

- Відстеження змін та взаємодія з адміністратором:

Система відстежує зміни в поведінковому профілі і може повідомити адміністратора про виявлені аномалії або зміни, які можуть вимагати уваги або додаткового аналізу.

- Ідентифікація ризикових сценаріїв:

На основі оновленого профілю система може переоцінити сценарії ризиків і виявити нові потенційні загрози.

- Використання змін у системі безпеки:

Якщо оновлення профілю вказує на нові фактори ризику, система може автоматично або за допомогою адміністратора застосувати відповідні заходи безпеки.

- Забезпечення адаптивності та ефективності:

Система відстежує ефективність свого оновленого поведінкового профілю і, за необхідності, коригує параметри, щоб забезпечити максимальну адаптивність і точність.

4) Висновки та попередження: Цей етап є завершальним для демо-блоку в роботі системи моніторингу та аналізу активності користувачів. На цьому етапі визначаються результати аналізу, приймаються рішення і, при необхідності, здійснюється оповіщення адміністратора або інших відповідальних осіб. Розглянемо цей етап детальніше:

- Аналіз результатів:

Система проводить фінальний аналіз активності користувачів, використовуючи оновлений поведінковий профіль і виявлені аномалії. Розглядаються закономірності, тенденції та будь-які невідповідності зі звичним поведінковим шаблоном користувача.

- Генерація висновків:

На основі аналізу система формує висновки про стан безпеки та активність користувача. Це може включати ідентифікацію потенційних загроз, виявлення аномалій та оцінку ризиків.

- Прийняття рішень:

На основі отриманих даних система приймає рішення про те, як діяти далі. Це може включати автоматичне реагування на виявлені загрози або вимагати втручання адміністратора для подальших дій.

- Повідомлення Адміністратора:

У разі виявлення аномалій або загроз система генерує повідомлення для

адміністратора або відповідальної особи. Це можуть бути повідомлення на електронну пошту, корпоративний месенджер або інші канали зв'язку.

- Подальші дії:

Сповіщення може містити рекомендації або конкретні кроки для подальших дій. Адміністратор може взяти ситуацію під контроль і вжити заходів для забезпечення безпеки системи.

- Документування та архівування:

Результати аналізу, висновки та рішення документуються для подальшого використання, аналізу та архівування. Це важливо для збереження історії безпеки, а також для можливості аналізу та вдосконалення системи.

Цей етап важливий для забезпечення безпеки системи та своєчасного і ефективного реагування на потенційні загрози.

4.2 Розробка та дослідження тестової програмної моделі для автоматизованої компіляції одиночних «демон-юнітів»

У цьому розділі розглядається розробка та дослідження тестової програмної моделі, спрямованої на автоматизовану компіляцію одиночних «демон-юнітів». Дана модель слугує експериментальною основою для визначення ефективності та функціональності автоматизованого процесу компіляції.

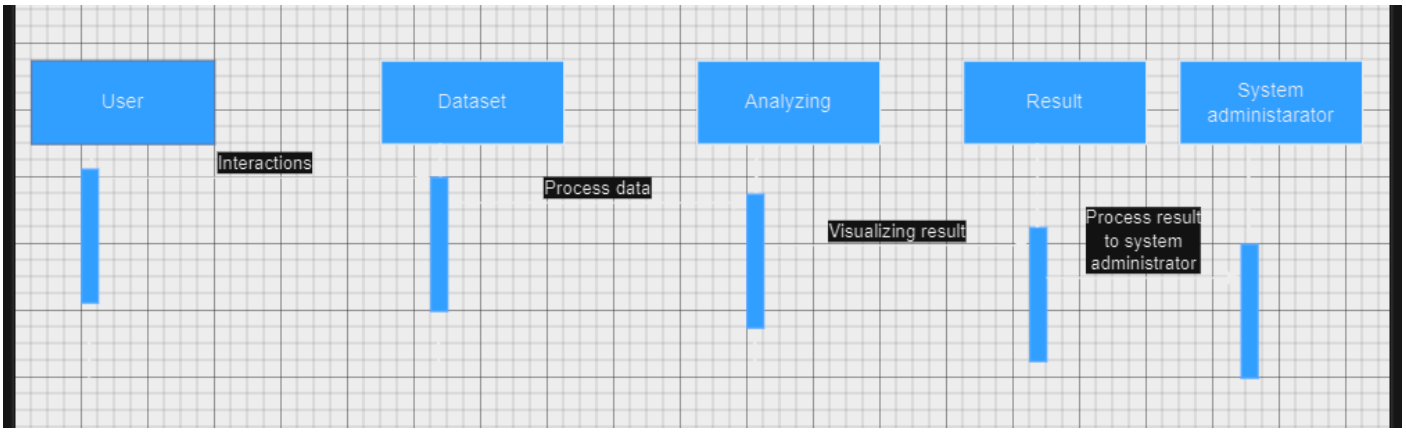

Рисунок 4.2 - Діаграма алгоритму

На рис 4.2 представлена діаграма алгоритму та, фактично, принцип роботи тестової моделі «демон-юніту». «Демон-юніт» збирає усю необхідну інформацію з дій користувача. Вся інформація зберігається у зашифрованому датасеті з ключем. Після чого аналізується та

системному адміністратору видається результат роботи «демон-юніту».

На рис 4.3 представлена діаграма класів даного алгоритму. Основним класом є CAnalyzeAllData, в якому відбувається апробація та аналіз результатів дій користувача. також дуже важливим є клас CWatcher у якому і відбувається компіляція та запуск «демон-юніту» для збору інформації. У класі CNetworkActivity виконується етап збору усієї мережевої інформації користувача. CGetSystemInfo відповідає за збір усієї інформації з пристрою користувача. CResults відповідає за виведення результатів. Може бути розширеним додаванням функцій для побудування діаграм та більш юзер-френдлі інтерфейсу.

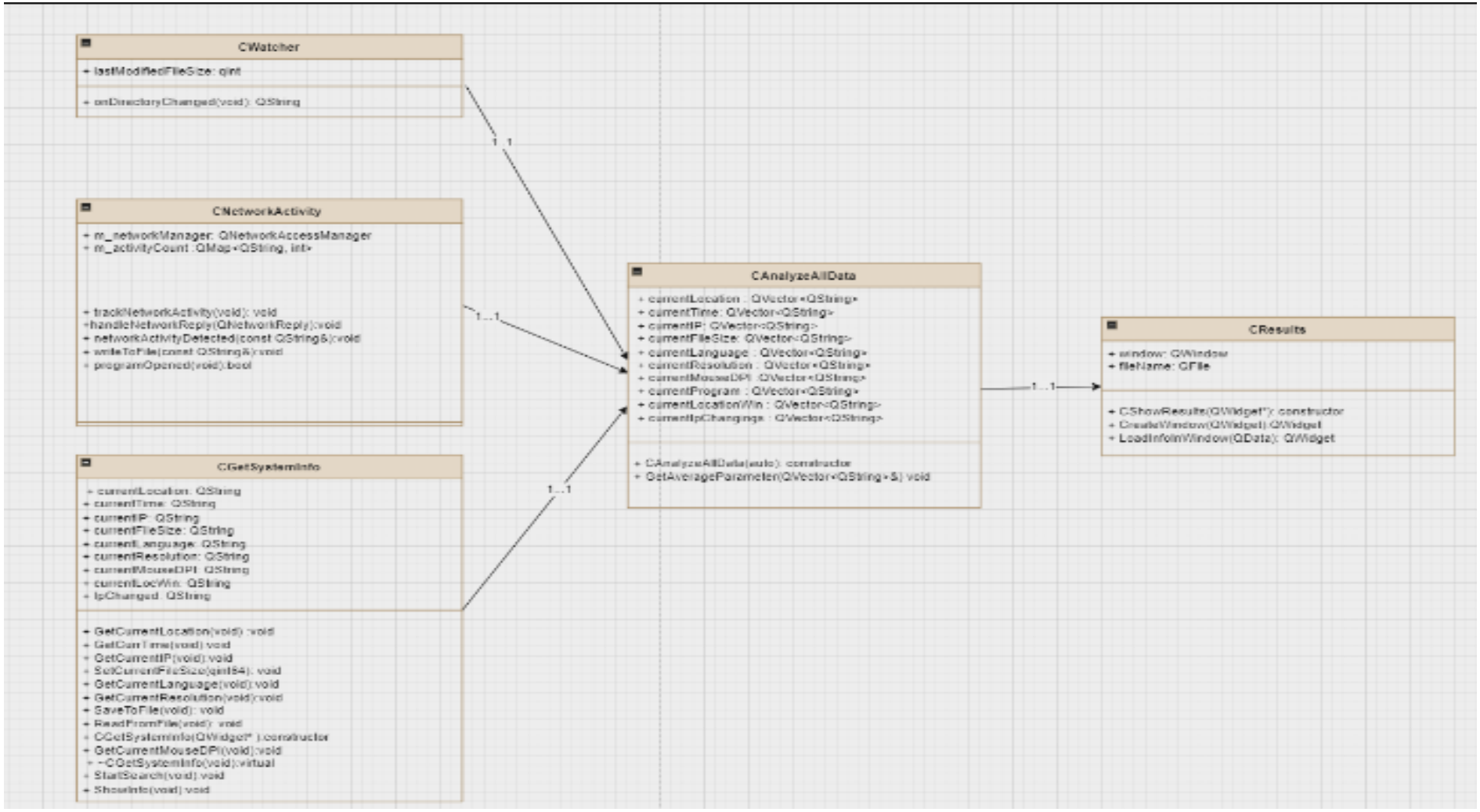

Рисунок 4.3 - Діаграма класів

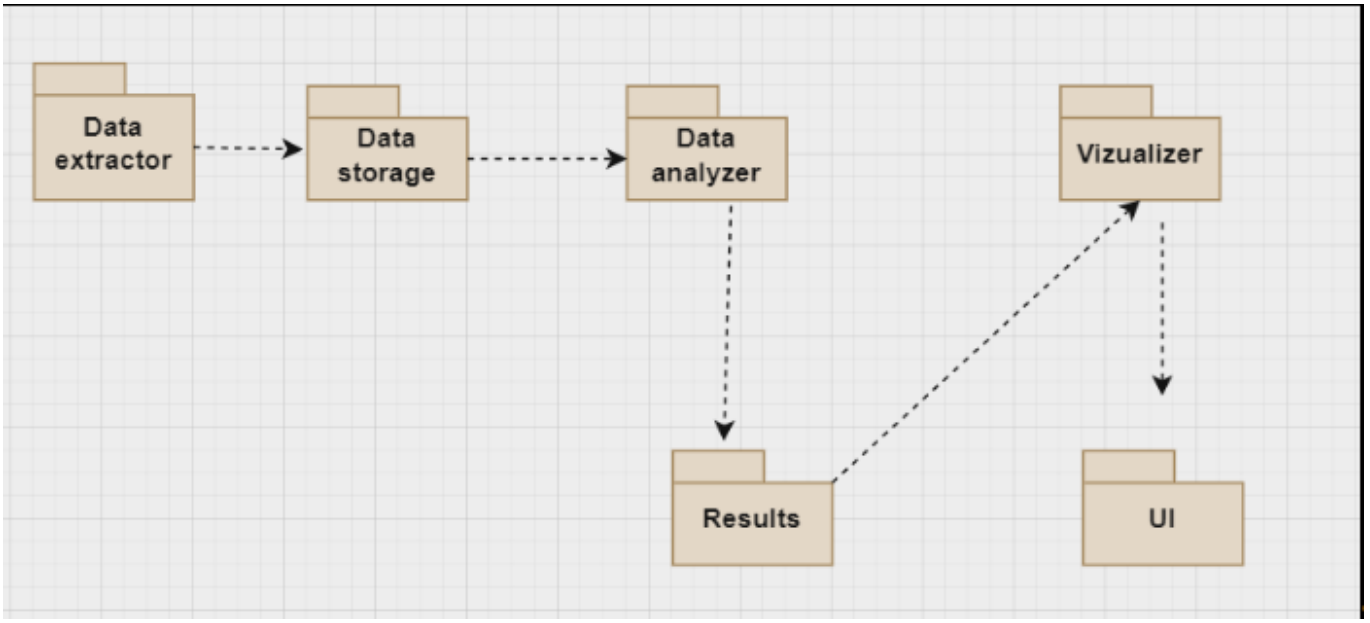

Рисунок 4.4 - Діаграма пакетів

На рис 4.4 представлена діаграма пакетів яка по суті показує, що даний алгоритм є цілком модульним та при можливості або бажанні компоненти можуть бути легко замінені на аналоги. Основним тут є модуль Data extractor, який включає в себе збір усієї необхідної інформації про користувача.

4.3 Відпрацювання питань користувальницького інтерфейсу застосунку та автоматизації складання параметричної структури персоніфікованих «демон-юнітів»

Опрацювання питань користувацького інтерфейсу додатку та автоматизація створення параметричної структури персоналізованих «демон-юнітів» є важливим етапом у розробці таких систем. Основними аспектами цього процесу є наступні:

1) Проектування інтерфейсу користувача :

Інтерфейс повинен бути зрозумілим для системного адміністратора. Він також повинен бути мінімалістичним, через можливе високе навантаження на сервер, на якому працює системний адміністратор.

2) Структура додатку:

Додаток повинен бути добре структурованим - компоненти додатку повинні легко замінюватися на новіші. Видалення та заміна компонента додатку не повинно переривати роботу додатку.

3) Введення параметрів «демон-юнітів»:

Повинні бути прапорці, контекстні меню та поля для встановлення параметрів. Всі параметри повинні бути доступними для редагування.

4) Автоматизація компіляції структури:

Додаток повинен містити автоматизований процес побудови параметричної структури «демон-юнітів».

5) Відстеження змін та історія конфігурації:

Повинна бути можливість відновлення попередніх налаштувань, відкоту до попередньої версії та модуль відстеження змін.

6) Валідація та верифікація даних:

Будь-який параметр повинен мати валідацію вхідних даних. Це має усунути потенційні помилки в конфігурації параметрів і дозволити системному адміністратору «демон-юніта» негайно побачити помилку.

7) Візуалізація та графіки:

Інтерфейс користувача повинен мати кнопку для відображення візуалізованих даних на екрані. Це дуже важливий етап, оскільки інформація краще сприймається і піддається людському аналізу через візуалізацію змін.

8) Контроль доступу:

Повинна бути система контролю доступу до додатку. Вся інформація, яка буде міститися в цьому додатку, не повинна випадково або цілеспрямовано потрапити до особи, яка не має доступу.

9) Документація та підтримка:

Все, що міститься в цій заявці, повинно бути задокументовано, а на випадок виникнення проблем повинні бути надані прості у виконанні інструкції. Повинен бути перелік типових проблем та способи їх вирішення.

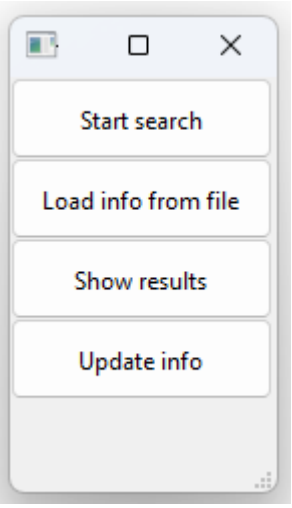

Рисунок 4.5- Приклад програмного інтерфейсу

4.4 Формування пропозицій, щодо структури та функціональних завдань алгоритму автоматизованої компіляції виконавчого модуля для одиночних «демон-юнітів», з урахуванням особливостей сучасних ІС

Формування пропозицій щодо структури та функціональних завдань алгоритму автоматизованої компіляції для окремих «демон-юнітів» включає розробку ефективного та надійного механізму перетворення вихідних даних виконавчого модуля для коректного та оптимального виконання функціональних завдань. Розглянемо основні аспекти та етапи:

1) Визначення функціональних завдань:

Конкретні функціональні завдання, які повинен виконувати модуль «демон-юніта». Це може бути збір та аналіз телеметричних даних, виявлення аномалій, моніторинг та управління ресурсами тощо.

2) Структура вихідних даних:

Структура вихідних даних виконавчого модуля. Дуже важливий момент для подальшої інтеграції та взаємодії з іншими системами та компонентами.

3) Мова програмування:

Мова програмування повинна бути виконана безпосередньо в машинному коді, для швидкості роботи.

4) Оптимізація та підтримка апаратного забезпечення:

Функції оптимізації виконуваного коду для максимізації ефективності апаратного

забезпечення.

5) Безпека:

Аспекти безпеки під час компіляції. відповідні методи та алгоритми повинні використовуватися для захисту виконуваного коду від атак та несанкціонованих змін.

6) Автоматизовані тести та валідація:

Розробіть автоматизовані тести для перевірки часу виконання. Це допоможе перевірити правильність роботи та виявити можливі помилки та недоліки.

7) Оновлення та розгортання:

Розробіть механізми оновлення виконавчого модуля. Розгляньте можливість автоматичного розгортання та моніторингу нових версій.

8) Документація та підтримка:

Надайте детальну технічну документацію для адміністраторів та розробників. Розробити систему підтримки та вирішення проблем для забезпечення ефективної взаємодії з користувачами.

9) Взаємодія з іншими «демон-юнітами»:

Розгляньте можливості взаємодії та координації з іншими «демон-юнітами» в корпоративному середовищі. Це може включати обмін інформацією та спільні дії для підвищення загальної ефективності.

ВИСНОВКИ

В результаті проведення роботи були досліджені технології та методи аналізу мережевої активності користувачів, включаючи дані телеметрії, поведінкові параметри та пошукова рефлексія. В процесі досліджень синтезовано структуру алгоритму вилучення даних поведінкового профілю користувачів, зокрема, розроблено механізми отримання та обробки цих параметрів в системі Windows.

У розділі, присвяченому синтезу алгоритму компіляції окремих «демон юнітів», розглянуто етапи та принципи формування програмної моделі з урахуванням потреб корпоративного середовища. Особливий акцент зроблено на аспектах безпеки, оптимізації та можливості взаємодії з іншими «демон-юнітами».

В ході моделювання підтверджено припущення, що етапи екстракції, накопичення та узагальнення даних про мережеву активність користувачів сучасних ІС, виступають в якості дуже інформативної основи для подальшого синтезу спеціалізованих алгоритмів компіляції одиничних або групових «демон-юнітів» із заданими поведінковими властивостями. Неперервність та таргетоване вилучення потрібних даних, щодо мережевих подій й характерних поведінкових профілів користувачів, дозволяє суттєво підвищити якість контролю мережевої взаємодії та забезпечити більш точніше виявлення аномалій та ознак існуючих загроз безпеки.

Зроблено висновок, що комплексна реалізація інструментів мережевого спостереження є ключовим аспектом для успішного вирішення завдань на наступних етапах створення систем для емуляції мережевих співтовариств з потрібним рівнем їх (VR бот-екосистем) кластерізації. При цьому, складання відповідних систем із «демон-юнітів» з потрібними властивостями, може мати дуже широкий спектр функціональних завдань. Наприклад, це створює необхідну основу для подальшого синтезу спеціалізованих безпекових платформ, що віртуалізують задані процеси та/чи елементи мережевої інфраструктури сучасних й майбутніх

(ІС).

За результатами роботи отримано програмну модель, яка інтегрує аналіз та синтез параметрів мережевої поведінки одиночних користувачів, що сприяє ефективному контролю функціонування корпоративної мережі та моніторингу поточного стану безпеки в частині визначення поведінкових колізій та недопущення витоку чутливих даних. Продемонстровано, що розроблені алгоритми та програмні моделі, можуть бути успішно використані в умовах роботи сучасних ІС. Результати тестування діючих прототипів програмної моделі алгоритмів екстрактору мережевих даних та синтезу поведінкового профілю одиночного «демон-юніту», порівняно з існуючими рішеннями, підтверджують їх застосовність. Подальші напрямки досліджень повинні бути спрямовані за двома основними векторами робіт: створення групових «демон-юнітів» (тобто, т.з. «бот-ферм»); синтезі повністю автономних алгоритмів для синтезу одиночних «демон-юнітів» із заданими властивостями.

Таким чином, проведені дослідження та моделювання основних процесів в межах вищезазначених напрямів робіт, створюють необхідну основу для подальшого синтезу спеціалізованих безпекових платформ, що віртуалізують потрібні процеси та/чи елементи мережевої інфраструктури, як сучасних, так і майбутніх ІС.

СПИСОК ДЖЕРЕЛ ПОСИЛАНЬ

1. NIST. Blockchain Technology Overview. URL: https://nvlpubs.nist.gov/nistpubs/ir/2018/nist.ir.8202.pdf (дата звернення: 8.05.2023)
2. Структура Меркла-Дамгора. URL: https://ru.wikipedia.org/wiki/Структура_Меркла_—_Дамгора (дата звернення: 12.05.2023)
3. The History & Future of Blockchain Technology. URL: https://www.linkedin.com/pulse/history-future-blockchain-technology-the-coin-times (дата звернення: 31.05.2023)
4. Dawit Andargachew Asmare. Blockchain Technology: Understanding its Meaning, Architecture, and Diverse Applications. URL: https://www.researchgate.net/figure/Stuart-Haber-and-W-Scott-Stornetta_fig3_373236964 (дата звернення: 31.05.2023)
5. W. Diffie, M. E. Hellman. New Directions in Cryptography. URL: https://www-ee.stanford.edu/~hellman/publications/24.pdf (дата звернення: 2.06.2023)
6. J. Pan al. Signed (Group) Diffie-Hellman Key Exchange with Tight Security. URL: https://eprint.iacr.org/2021/607.pdf (дата звернення: 10.06.2023)
7. S. Haber, W. Stornetta. How To Time-Stamp a Digital Document. URL: https://link.springer.com/content/pdf/10.1007/BF00196791.pdf (дата звернення: 15.07.2023)
8. L. Lamport. Paxos Made Simple. URL: https://lamport.azurewebsites.net/pubs/paxos-simple.pdf (дата звернення: 15.07.2023)
9. D. Ongaro, J. Ousterhout. In Search of an Understandable Consensus Algorithm (Extended Version). URL: https://raft.github.io/raft.pdf (дата звернення: 15.07.2023)
10. Saqib I. Comparison of different firewalls performance in a virtual for cloud data center. Journal of advancement in computing. 2023. Vol. 1. P. 21–28. URL: https://journalsriuf.com/index.php/JAC/article/view/49/59 (дата звернення:

19.09.2023)

11. Dieterich A. Evaluation of persistence methods used by malware on microsoft windows systems. Proceedings of the 9th international conference on information systems security and privacy (ICISSP 2023). 2023. P. 552–559. URL: https://www.scitepress.org/Papers/2023/117102/117102.pdf (дата звернення: 02.07.2023)
12. Колованова Є., Малахов С., Чорна Т. Передумови та основні складови з протидії доксінгу персональних даних. The 27th International scientific and practical conference "Trends of young scientists regarding the development of science", м. Едмонтон, 11–14 лип. 2023 р. Едмонтон, 2023. С. 194. URL: https://isg-konf.com/wp-content/uploads/2023/07/TRENDS-OF-YOUNG-SCIENTISTS-REGARDING-THE-DEVELOPMENT-OF-SCIENCE.pdf (дата звернення: 29.09.2023)
13. Михайленко Д., Чорна Т., Малахов С. Використання можливостей AI при реалізації Static та Dynamic Honeypot для покращення параметрів захисту інформаційних ресурсів. Використання можливостей AI при реалізації Static та Dynamic Honeypot для покращення параметрів захисту інформаційних ресурсів : матеріали IV Міжнар. наук. конф., м. Суми, 7 жовт. 2022 р. Суми, 2022. С. 54–57. (дата звернення: 01.10.2023)
14. Гайкова В., Малахов С. Аналіз факторів і умов реалізації кібербулінгу з урахуванням можливостей сучасних інформаційних систем. Комп'ютерні науки та кібербезпека. 2021. Т. 1, № 1. С. 50–59. URL: https://doi.org/10.26565/2519-2310-2021-1-04. (дата звернення: 13.07.2023)
15. Rachana PatelSanskruti, PatelSanskruti Patel. A Comprehensive Study of Applying Convolutional Neural Network for Computer Vision // International Journal of Advanced Science and Technology 6(6):2161-2174. Jan 2020. URL: https://www.researchgate.net/publication/344121826_A_Comprehensive_Study_of_Applying_Convolutional_Neural_Network_for_Computer_Vision. (дата звернення: 18.10.2023)
16. Andrea F.Abate, Carmen Bisogni, Carmen Bisogni, Aniello Castiglione. Head Pose

Estimation: An Extensive Survey on Recent Techniques and Applications // Pattern Recognition 127(4):108591. Feb 2022. URL: https://www.researchgate.net/publication/358647531_Head_Pose_Estimation_An_Extensive_Survey_on_Recent_Techniques_and_Applications. (дата звернення: 14.09.2023)

17. Mark Sandler et.al. MobileNetV2: Inverted Residuals and Linear Bottlenecks // CVPR, 13 Jan 2018. URL: https://arxiv.org/abs/1801.04381. (дата звернення: 20.10.2023)

ДОДАТОК А

ЛІСТИНГ КОДУ

```
//CAnalyzeAllData.h
#include <QString>
#include <QVector>


class CAnalyzeAllData
{
public:
    CAnalyzeAllData(auto& allData);
private:
    void GetAverageParameter(QVector<QString>& parameter);
private:
    QVector<QString> currentLocation;
    QVector<QString> currentTime;
    QVector<QString> currentIP;
    QVector<QString> currentFileSize;
    QVector<QString> currentLanguage;
    QVector<QString> currentResolution;
    QVector<QString> currentMouseDPI;
    QVector<QString> currentProgram;
    QVector<QString> currentLocationWin;
    QVector<QString> currentIpChangings;
};
//CWatcher.h
```

```
#include <QObject>
#include <QString>
#include <QFileInfo>
#include <QFileSystemWatcher>
#include <QDir>
class CWatcher : public QObject {
    Q_OBJECT
private:
    qint64 lastModifiedFileSize;
public:
    CWatcher(QObject* parent = nullptr);
    const QString GetFileSize();

private slots:
    void onDirectoryChanged(const QString& path);
};
//CGettingSystemInfo.h
#pragma once
#include <iostream>
#include <array>
#include <wchar.h>
#include <tchar.h>
#include <Windows.h>
#include <winnls.h>
#include <winsock.h>
#include <ctime>
#include <cmath>
#include <chrono>
#include <fstream>
```

```
#include <QString>
#include <QWidget>
#include "cwatcher.h"
struct IPv4
{
	unsigned char b1, b2, b3, b4;
};

class CGetSystemInfo : public QWidget
{
   Q_OBJECT
private:
   QString currentLocation;
   QString currentTime;
   QString currentIP;
   QString currentFileSize;
   QString currentLanguage;
   QString currentResolution;
   QString currentMouseDPI;
   QString currentLocWin;
   QString IpChanged;
private:
	void GetCurrentLocation();
	void GetCurrTime();
   void GetCurrentIP();
   void SetCurrentFileSize(qint64 filesize);
	void GetCurrentLanguage();
	void GetCurrentResolution();
	void SaveToFile();
```

```
    void ReadFromFile();
public:
    CGetSystemInfo(QWidget* parent = 0);
    void GetCurrentMouseDPI();
    virtual ~CGetSystemInfo();
public slots:
    void StartSearch();
    void ShowInfo();
};
//CNetworkActivity.h
#include <QObject>
#include <QMap>
#include <QNetworkAccessManager>

class CNetworkActivity : public QObject
{
    Q_OBJECT
public:
    explicit CNetworkActivity(QObject *parent = nullptr);
    void writeToFile(QString filename);
    void telegramOpened();

signals:
    void networkActivityDetected(const QString &applicationName);

private slots:
    void trackNetworkActivity();
    void handleNetworkReply(QNetworkReply *reply);
```

```
private:
    QNetworkAccessManager *m_networkManager;
    QMap<QString, int> m_activityCount;
};

//CWatcher.cpp
#include "cwatcher.h"

CWatcher::CWatcher(QObject* parent) : QObject(parent)
{
    QFileSystemWatcher* watcher = new QFileSystemWatcher(this);
    watcher->addPath(QDir::homePath());

    connect(watcher, &QFileSystemWatcher::directoryChanged, this, &CWatcher::onDirectoryChanged);
}

const QString CWatcher::GetFileSize()
{
    QFile file("E:/projects/Diplom4_0/FileSize.txt");
    file.open(QIODevice::Append | QIODevice::Text);
    QTextStream out(&file);
    if (!file.isOpen())
    {
        qDebug() << "Error, file dont created\n";
    }
    else
    {
        out << lastModifiedFileSize << "\n";
    }
```

```
    file.close();
}

void CWatcher::onDirectoryChanged(const QString &path)
{
    QDir directory(path);
    directory.setFilter(QDir::Files | QDir::NoDotAndDotDot);
    QFileInfoList fileInfoList = directory.entryInfoList();
    if (!fileInfoList.isEmpty()) {
        QFileInfo lastModifiedFile = fileInfoList.last();
        lastModifiedFileSize = lastModifiedFile.size();
    }
}

//CNetworkActivity.cpp
#include "cnetworkactivity.h"
#include <QTimer>
#include <QFile>
#include <qprocess.h>

CNetworkActivity::CNetworkActivity(QObject *parent)
    : QObject(parent)
{
    m_networkManager = new QNetworkAccessManager(this);
    connect(m_networkManager, &QNetworkAccessManager::finished, this,
&CNetworkActivity::handleNetworkReply);

    // Инициализация частоты обновления (в миллисекундах)
    int updateFrequency = 1000; // Например, обновляем каждые 1000 миллисекунд (1 секунда)
```

```
    QTimer *timer = new QTimer(this);
    connect(timer, &QTimer::timeout, this, &CNetworkActivity::trackNetworkActivity);
    timer->start(updateFrequency);
}

void CNetworkActivity::writeToFile(QString filename)
{
    QFile file("E:/projects/Diplom4_0/Network.txt");
    file.open(QIODevice::Append | QIODevice::Text);
    QTextStream out(&file);
    if (!file.isOpen())
    {
        qDebug() << "Error, file dont created\n";
    }
    else
    {
        out << filename << "\n";
    }
    file.close();
}

void CNetworkActivity::telegramOpened()
{
    QString telegramPath = "C:\\Users\\Сергей\\AppData\\Roaming\\Microsoft\\Windows\\Start
Menu\\Programs\\Telegram.exe";
    QProcess telegramProcess;

    // Запуск процесса Telegram
    telegramProcess.start(telegramPath);
```

```
    // Ожидание завершения процесса
    if (telegramProcess.waitForFinished(-1)) {
        writeToFile("TelegramOpened");
    }
}

void CNetworkActivity::trackNetworkActivity()
{
    // Проверка частоты активности сети по приложениям
    for (const QString &appName : m_activityCount.keys()) {
        int count = m_activityCount.value(appName);
        if (count > 0) {
            emit networkActivityDetected(appName);
        }
    }

    m_activityCount.clear();
}
//GettingSystemInfo.cpp
#include "GettingSystemInfo.h"
#include <QDate>
#include <QHostAddress>
#include <QNetworkInterface>
#include <qfile.h>
#include <QFileSystemWatcher>
#include <QDebug>
#include <QTextStream>
#include <QString>
```

```
#include <QGeoPositionInfoSource>
#include <QGeoPositionInfo>
#include <QApplication>
#include <QScreen>
#include <QDir>

void CGetSystemInfo::GetCurrentLocation()
{
    QGeoPositionInfoSource *source = QGeoPositionInfoSource::createSource("Provider", this);
    if (source->lastKnownPosition().coordinate().toString().isEmpty())
    {
        currentLocation = "No port found ";
    }
    else
    {
        currentLocation = source->lastKnownPosition().coordinate().toString();
    }
}

void CGetSystemInfo::GetCurrTime()
{
    QDateTime current = QDateTime::currentDateTime();
    currentTime = current.toString();
}

void CGetSystemInfo::GetCurrentIP()
{
    const QHostAddress &localhost = QHostAddress(QHostAddress::LocalHost);
    for (const QHostAddress &address: QNetworkInterface::allAddresses()) {
```

```
        if (address.protocol() == QAbstractSocket::IPv4Protocol && address != localhost)
            this->currentIP = address.toString();
    }
}

void CGetSystemInfo::GetCurrentLanguage()
{
    // Get the user's selected language
    // Defaults to the system installed language if not using MUI.
    LANGID langID = GetUserDefaultUILanguage();

    // Convert LANGID to a RFC 4646 language tag (per navigator.language)
    int langSize = GetLocaleInfo(langID, LOCALE_SISO639LANGNAME, NULL, 0);
    int countrySize = GetLocaleInfo(langID, LOCALE_SISO3166CTRYNAME, NULL, 0);

    wchar_t* lang = new wchar_t[langSize + countrySize + 1];
    wchar_t* country = new wchar_t[countrySize];

    GetLocaleInfo(langID, LOCALE_SISO639LANGNAME, lang, langSize);
    GetLocaleInfo(langID, LOCALE_SISO3166CTRYNAME, country, countrySize);

    // add hyphen
    wcscat(wcscat(lang, L"-"), country);
    std::wstring locale(lang);

    delete[] lang;
    delete[] country;
    currentLanguage = QString::fromStdWString(locale);
}
```

```
void CGetSystemInfo::GetCurrentResolution()
{
    int width, height;
    width = GetSystemMetrics(SM_CXSCREEN);
    height = GetSystemMetrics(SM_CYSCREEN);
    auto s = std::to_string(width) + "x" + std::to_string(height);
    currentResolution = QString::fromStdString(s);
}

void CGetSystemInfo::SaveToFile()
{
    QFile file("E:/projects/Diplom4_0/Info.txt");
    file.open(QIODevice::Append | QIODevice::Text);
    QTextStream out(&file);
    if (!file.isOpen())
    {
        qDebug() << "Error, file dont created\n";
    }
    else
    {
        out << currentLocation  << " " << currentIP << " " << currentResolution  << " " << currentTime  <<
" " << currentLanguage << " " << currentMouseDPI << "\n";
    }
    file.close();
}

void CGetSystemInfo::ReadFromFile()
{
```

```
    QFile file("E:/projects/Diplom4_0/Info.txt");


    if (file.open(QIODevice::ReadOnly))
    {
        auto readed = file.readAll();
    }
}


CGetSystemInfo::CGetSystemInfo(QWidget *parent) : QWidget(parent)
{
}




void CGetSystemInfo::GetCurrentMouseDPI()
{
    // Get the DPI of the screen
    qreal dpiX = logicalDpiX();
    currentMouseDPI = QString::number(dpiX * 5);
}


CGetSystemInfo::~CGetSystemInfo()
{


}


void CGetSystemInfo::StartSearch()
{
    this->GetCurrentLocation();
```

```
    this->GetCurrentIP();
    this->GetCurrentResolution();
    this->GetCurrTime();
    this->GetCurrentLanguage();
    this->GetCurrentMouseDPI();
    SaveToFile();
}
//main.cpp
#include "mainwindow.h"

#include <QApplication>
#include <QPushButton>
#include <QFileSystemWatcher>
#include "GettingSystemInfo.h"
/#include "cnetworkactivity.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    CGetSystemInfo info;
    CNetworkActivity networkActivityTracker;
    QPushButton* startSearch = w.window()->findChild<QPushButton*>("StartSearch");
    QPushButton* showInfo = w.window()->findChild<QPushButton*>("ShowInfo");
    QPushButton* updateInfo = w.window()->findChild<QPushButton*>("UpdateInfo");
    QObject::connect(startSearch, &QPushButton::clicked,&info,&CGetSystemInfo::StartSearch);
    QObject::connect(showInfo, &QPushButton::clicked,&info,&CGetSystemInfo::ShowInfo);
    QObject::connect(&networkActivityTracker,
&CNetworkActivity::networkActivityDetected,&CNetworkActivity::writeToFile);
```

```
    return a.exec();

}
```