Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

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

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітній ступень бакалавр

освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: Програмний засіб для підвищення рівня анонімності користувача в глобальній мережі

Виконавець: студентка IV курсу, групи КБ-41

_____ **Аліна КИСЕЛЬОВА**
(підпис) (ім'я прізвище)

	Прізвище, ініціали	**Підпис**
Керівник	Сергій ДАКОВ	

Нормоконтроль	Інна МИХАЛЬЧУК	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)
освітньої програми Кібербезпека
(назва освітньої програми)

Студентці **КБ-41** **Кисельовій Аліні Павлівні**
(група) (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Програмний засіб для підвищення рівня анонімності користувача в глобальній мережі

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Програмний засіб для підвищення рівня анонімності користувача в Інтернеті.

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Ознайомитись із теоретичними дослідженнями в галузі вебтрекінгу, методами відстеження, провести порівняння наявних методів захисту. Розробити алгоритм, що буде перешкоджати відстеженню користувача та підвищувати рівень його анонімності в мережі, провести тестування і довести ефективність розробленого програмного засобу.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розробка програмного засобу для підвищення рівня анонімності користувача в мережі Інтернет.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

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

Завдання прийняла до виконання _____ (підпис) Аліна КИСЕЛЬОВА (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 28.11.2022	*виконано*
2	Дослідження та аналіз джерел за темою дослідження	29.11.2022 – 15.01.2023	*виконано*
3	Вивчення нормативно-правової бази в галузі вебтрекінгу	16.01.2023 – 29.01.2023	*виконано*
4	Аналіз функціоналу та тестування існуючих програмних рішень для захисту від вебтрекінгу	30.01.2023 – 01.03.2023	*виконано*
5	Планування реалізації практичної частини	02.03.2023 – 14.03.2023	*виконано*
6	Розробка власного алгоритму ПЗ для запобігання вебтрекінгу в Google Chrome	15.03.2023 – 25.04.2023	*виконано*
7	Тестування розробленого ПЗ	26.04.2023 – 15.05.2023	*виконано*
8	Оформлення пояснювальної записки	16.05.2023 – 01.06.2023	*виконано*
9	Підготовка до захисту	02.06.2023 – 12.06.2023	*виконано*

Завдання видав _____ (підпис) Сергій ДАКОВ (ініціали, прізвище)

Завдання прийняла до виконання _____ (підпис) Аліна КИСЕЛЬОВА (ініціали, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, має 80 сторінок основного тексту, включає в себе зміст, вступ, три розділи роботи, висновки та список джерел. Робота містить 2 додатки із загальною кількістю сторінок 7. У пояснювальній записці кваліфікаційної роботи міститься 68 рисунків і 4 таблиці. Список використаних джерел містить 48 найменувань і займає 4 сторінки.

Мета роботи: розробка програмного засобу захисту користувачів від відстеження в браузері для підвищення рівня анонімності в глобальній мережі.

Для досягнення мети планується провести аналіз існуючих сучасних методів захисту проти вебтрекінгу; розробити власний алгоритм, який буде ефективним у боротьбі з відстеженням в Інтернеті; провести тестування розробленого програмного засобу та довести його ефективність.

Об'єкт дослідження: процес захисту даних користувачів в Інтернет-просторі для підвищення рівня анонімності користувача в глобальній мережі.

Предмет дослідження: методи та засоби для підвищення рівня анонімності користувача в глобальній мережі.

Методи дослідження: аналіз, порівняння, проєктування, тестування.

Практична цінність кваліфікаційної роботи полягає в розробці програмного засобу для підвищення рівня анонімності користувача в мережі Інтернет на базі спуфінгу цифрових відбитків браузера.

Ключові слова: вебтрекінг, відстеження в браузері, цифровий відбиток, унікальний ідентифікатор користувача, фінгерпринтинг, анонімність, приватність, конфіденційність, вебпереглядач.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AES	–	Advanced Encryption Standard
API	–	Application Programming Interface
CCPA	–	California Consumer Privacy Act
DNT	–	Do Not Track
DoH	–	DNS-over-HTTPS
ETP	–	Enhanced Tracking Protection
GDPR	–	The General Data Protection Regulation
HTTP	–	HyperText Transfer Protocol
HTTPS	–	HyperText Transfer Protocol Secure
ISP	–	Internet Service Provider
ITP	–	Intelligent Tracking Prevention
TOR	–	The Onion Router
UA	–	User-Agent
URL	–	Uniform Resource Locator
VPN	–	Virtual Private Network
ПЗ	–	Програмне забезпечення

ВСТУП

Актуальність даної роботи базується на тому, що вебвідстеження стало поширеною практикою в Інтернеті, де компанії та вебсайти відстежують поведінку користувачів для різних цілей, таких як цільова реклама, аналітика, персоналізація тощо. Ця практика викликає занепокоєння щодо конфіденційності та анонімності користувачів і безпеки даних, оскільки конфіденційна інформація може стати доступною для небажаних третіх сторін та особа користувача може бути однозначно ідентифікована.

З розвитком технологій відстеження традиційних засобів захисту конфіденційності, таких як блокувальники реклами та трекерів, уже недостатньо для захисту конфіденційності користувачів. Такі методи, як цифрові відбитки у вебпереглядачі, коли вебсайти можуть ідентифікувати користувачів на основі унікальних конфігурацій браузера, ускладнюють збереження анонімності в Інтернеті.

Саме тому розробка програмного засобу для захисту від вебвідстеження, спрямована на спуфінг цифрових відбитків, стає дуже актуальною. Такий інструмент може допомогти користувачам зберігати конфіденційність і анонімність в Інтернеті, блокуючи або маскуючи технології відстеження, а також може завадити вебсайтам збирати конфіденційну інформацію про користувачів, зокрема їх історію вебперегляду, місцезнаходження та інформацію про пристрій тощо. Програмний засіб, базований на спуфінгу цифрових відбитків, може допомогти користувачам захистити свою анонімність в мережі Інтернет шляхом генерування випадкових значень для певних параметрів, таких як тип браузера, версія операційної системи, розмір екрану тощо, що робить унікальний ідентифікатор користувача менш точним та складнішим для встановлення. Це може підвищити рівень анонімності користувача та знизити ймовірність його відслідковування.

Як результат, описаний програмний засіб має на меті надавати користувачам можливість підвищити рівень анонімності в Інтернеті та захистити конфіденційну інформацію від небажаного відстеження та викриття.

Тому метою роботи є розробка програмного засобу захисту користувачів від відстеження в браузері для підвищення рівня анонімності в глобальній мережі. Для досягнення мети планується провести аналіз існуючих сучасних методів захисту проти вебтрекінгу; розробити власний алгоритм, який буде ефективним у боротьбі з відстеженням в Інтернеті; провести тестування розробленого програмного засобу та довести його ефективність.

Об'єктом дослідження є процес захисту даних користувачів в Інтернет-просторі для підвищення рівня анонімності користувача в глобальній мережі.

Предметом дослідження є методи та засоби для підвищення рівня анонімності користувача в глобальній мережі.

Методами дослідження є аналіз, порівняння, проектування, тестування.

Практична цінність кваліфікаційної роботи полягає в розробці програмного засобу для підвищення рівня анонімності користувача в мережі Інтернет на базі спуфінгу цифрових відбитків браузера.

Апробація результатів роботи. Основні результати кваліфікаційної роботи представлялися на V Міжнародній науково-практичній конференції "Проблеми кібербезпеки інформаційно-телекомунікаційних систем" (PCSITS-2022), VI Міжнародній науково-практичній конференції "Проблеми кібербезпеки інформаційно-телекомунікаційних систем" (PCSITS-2023) у матеріалах наукових конференцій.

РОЗДІЛ 1
МЕТОДИ ВЕБТРЕКІНГУ ТА ЗАХИСТ ВІД НИХ

1.1 Вебтрекінг як метод збору інформації в Інтернеті

Вебтрекінг (з англ. «web tracking» - вебвідстеження, стеження в Інтернеті) – це метод збору інформації про користувачів Інтернету з метою стеження за їхніми діями, профілювання і спостереження за їхніми звичками та інтересами, що може виконуватись безпосередньо власниками вебсайтів та сторонніми компаніями (рекламними, аналітичними, соціальними мережами тощо). Аналізуючи поведінку користувачів, оператори вебсайтів можуть надавати спеціалізований контент, який відображає інтереси відвідувачів, що може бути ефективною практикою для рекламодавців та інших сторін [1].

Різні види використання вебвідстеження включають персоналізовану рекламу на основі профілів користувачів, які створюються шляхом збору даних, таких як відвідані вебсайти, переглянуті відео, взаємодії в соціальних мережах і онлайн-транзакції. Наприклад, пошукові системи ведуть облік пошукових запитів користувачів, щоб пропонувати відповідні пошукові запити в майбутньому, а правоохоронні органи можуть використовувати вебвідстеження для розслідування окремих осіб і розкриття злочинів.

Крім того, вебаналітика зосереджується на продуктивності вебсайту, надаючи розуміння того, як вебсайт використовується та скільки часу користувачі проводять на конкретних сторінках. Цю інформацію можна використовувати, щоб визначити, який вміст є найбільш популярним серед користувачів.

Однак поруч із корисними функціями вебтрекінгу, є й негативні аспекти відстеження користувачів, такі як:

- Приватність і конфіденційність: вебтрекінг може порушувати приватність користувачів, оскільки він збирає інформацію про їхню активність без їхнього відома

чи згоди. Ця інформація може включати переглядені сторінки, пошукові запити, покупки онлайн, місцезнаходження та багато іншого.

- Продаж та обмін даними: зібрані дані про користувачів можуть бути продані стороннім компаніям, включаючи рекламні мережі, аналітичні компанії та іншим зацікавленим організаціям.
- Психологічний вплив: вебтрекінг може бути використаний для створення профілів користувачів і впливу на їхні переконання та поведінку. Це може включати використання персоналізованої політичної реклами, маніпуляцію інформацією та формування фільтрованого сприйняття світу.

Як основні методи вебтрекінгу виділяють відстеження IP-адреси, файли cookie, фінгерпринтинг в браузері, так звані «вебмаяки» тощо, огляд яких буде наведений у наступних пунктах.

1.2 Нормативно-правова база

У ЄС основними законодавчими актами щодо вебтрекінгу є The General Data Protection Regulation (GDPR) та ePrivacy Directive. GDPR вимагає від власників вебсайтів чітко вказувати, як вони відстежують користувачів, і дозволити їм відмовитися від вебтрекінгу. Вебсайти не повинні використовувати файли cookie або будь-який інший тип технології для відстеження поведінки користувачів без згоди. Сайт має надавати сповіщення та простий спосіб для користувачів відмовитися від відстеження. Власник вебсайту має переконатися, що він може підвердити надання користувачем згоди та навести пояснення, яким саме чином сайт використовуватиме ці дані. Частково дія GDPR поширюється і в США [2].

ePrivacy Directive передбачає, що користувачі мають бути поінформовані про використання файлів cookie та інших трекерів, що збирають їхні дані в Інтернеті. Крім того, ePrivacy Directive вимагає, щоб користувачі мали можливість контролювати використання цих трекерів, наприклад, шляхом встановлення налаштувань браузера, які дозволяють відхилити використання трекерів [3].

The California Consumer Privacy Act (CCPA) – закон штату Каліфорнія про конфіденційність споживачів, що застосовується в США для захисту особистих даних в Інтернеті, в якому затверджено, що якщо вебсайт збирає будь-які дані про користувачів, вебсайт повинен надати повідомлення та надати користувачам можливість відмовитися від відстеження. Вебсайти повинні надавати споживачам право знати, яку інформацію про них збирає компанія, і вимагати копію зібраної інформації [4].

В Україні питання регулювання вебтрекінгу є частиною загального законодавства про захист персональних даних, зокрема Закону України "Про захист персональних даних". Цей закон встановлює правила збору, зберігання, використання та поширення персональних даних, включаючи дані, що збираються через вебтрекінг [5]. Згідно зі статтею 6 Закону України "Про захист персональних даних", збір та обробка персональних даних можлива лише за умови наявності згоди суб'єкта персональних даних. Також, власники сайтів зобов'язані забезпечувати достатній рівень захисту персональних даних, зокрема за допомогою шифрування та захисту від несанкціонованого доступу. Додатково, в Україні діє Закон "Про рекламу", який містить правила реклами, включаючи персоналізовану рекламу, що може бути створена за допомогою вебтрекінгу. Зокрема, рекламні повідомлення повинні бути чітко відрізнятися від іншого контенту, а рекламні матеріали повинні бути достовірними та не містити недостовірної інформації [6]. Однак, в Україні поки що немає спеціалізованого законодавства, що б детально регулювало питання вебтрекінгу та його впливу на захист персональних даних користувачів. Тому зазначені правила та обмеження базуються на загальних принципах захисту персональних даних та рекламних правил.

1.3 Методи вебтрекінгу

Розглянемо деякі з сучасних методів вебвідстеження:

1. IP-адреса. Кожному пристрою, підключеному до Інтернету, призначається унікальний ідентифікатор – IP-адреса – для того, щоб пристрої мали змогу

обмінюватися даними один з одним в глобальній мережі. За допомогою відповідного програмного забезпечення на сервері вебсайту можна реєструвати IP-адреси відвідувачів, а також використовувати ці дані для визначення їх географічного розташування [7]. Оскільки місцезнаходження вказує на регіон або країну, сайт може автоматично робити налаштування для користувача, щоб ціни вказувалися в місцевій валюті, відображати асортимент доступних товарів, застосовувати мову регіону, а в деяких випадках запити або відповіді до певної країни будуть повністю заблоковані. Однак не завжди користувач зацікавлений в тому, щоб дані про його розташування розкривалися для сторонніх вебсайтів.

Для відстеження діяльності користувача на вебсторінці, його IP-адреса записується в журнал доступу до сайту, який містить інформацію про всі запити, зроблені користувачем. За допомогою інформації з цього журналу, вебсайт може відстежувати сторінки, які користувач переглянув, продукти, які він купив та інші дії на сайті.

Однак, варто зауважити, що метод вебтрекінгу, базований на IP-адресі, має свої обмеження. Зокрема, IP-адреса може бути загальною для декількох користувачів, наприклад, якщо вони знаходяться в одній мережі або користуються одним проксі-сервером. Користувачі Інтернету можуть обходити цензуру та геоблокування та захищати особисті дані та місцезнаходження, щоб залишатися анонімними в Інтернеті за допомогою, наприклад, VPN-з'єднання або використання проксі-серверів, детальніше про ці технології буде наведено в наступних пунктах.

2. Файли cookie. Метод вебтрекінгу, що базується на використанні файлів cookie, є одним з найпоширеніших методів збору інформації вебсайтами про користувачів. Cookies - це файли, які зберігаються на пристрої користувача вебсайтом, який він відвідав. При наступному відвідуванні цього вебсайту, браузер користувача надсилає збережені раніше файли на сервер вебсайту, де вони використовуються для відновлення певної інформації про користувача [8]. У процесі вебтрекінгу файли cookies використовуються для збору інформації різного виду, такої як перегляди сторінок, пошукові запити користувача, час відвідування та які дії на сайті він

виконав. Ця інформація може бути використана для ідентифікації користувача та здійснення персоналізованої реклами або для продажу третім сторонам.

Виділяють два типи файлів cookie – основні (first-party cookies) та сторонні (third-party cookies). Мета основних файлів cookie — розпізнати користувача та його вподобання, щоб можна було застосувати бажані налаштування. Сторонні файли cookie створюються іншими вебсайтами (третіми сторонами). Вони вставляють додатковий код відстеження, який може фіксувати онлайн-активність користувача. Аналітика на сайті стосується збору даних на поточному сайті. Він використовується для вимірювання багатьох аспектів взаємодії користувачів, включаючи кількість відвідувань користувача.

Алгоритм вебтрекінгу з використанням файлів cookies, можна описати декількома кроками:

- Коли користувач відвідує вебсайт, його браузер отримує файл cookie від цього сайту, який зберігається на його пристрої. Цей файл містить унікальний ідентифікатор користувача, який використовується для відстеження його діяльності на вебсайті, а також для наступних відвідувань цього сайту.
- Коли користувач повторно відвідує вебсайт, його браузер передає збережений файл cookie, що дозволяє сайту відрізнити його від інших відвідувачів та зберігати інформацію про його подальші дії на сайті. Інформація, яку збирається через файли cookie, може включати дані про переглянуті сторінки, час відвідування сайту, придбані товари, попередні пошукові запити, та іншу інформацію, що стосується діяльності користувача на сайті.
- Інформація, отримана з файлів cookie, може в подальшому використовуватися вебсайтом для налаштування відображення контенту, в якому користувач потенційно може бути зацікавлений, організації рекламних кампаній та для вивчення поведінки користувачів на сайті.

Ефективність вебтрекінгу, базованого на файлах cookies, може бути високою, оскільки цей метод дозволяє збирати інформацію про користувача, що дозволяє відображати спеціально налаштовані рекламні оголошення, відповідні до його інтересів. З іншого боку, ефективність вебтрекінгу також залежить від багатьох

чинників, таких як кількість інформації, яку користувачі надають про себе, а також як ця інформація використовується компаніями для налаштування рекламних кампаній. Крім того, користувач може або видаляти свої файли cookie, або налаштувати браузер, щоб вони не приймалися браузером, або використовувати спеціальні розширення для захисту від вебтрекінгу, які блокують cookies. У цьому випадку користувач може бути обмеженим у своїх діях на сайті, деякі функції або сервіси стануть недоступними, але інформація про його уподобання, пошукові запити тощо не потраплятиме до третіх сторін [8].

Існують сервіси, які слугують для того, щоб допомогти власникам вебсайтів організувати коректну роботу з файлами cookie та забезпечити відповідність з чинним законодавством про вебтрекінг щодо збору, зберігання та обробки даних користувачів. До таких інструментів належить Cookiebot, CookieYes, OneTrust тощо.

3. Фінгерпринтинг в браузері. Цифровий відбиток браузера – це набір інформації, пов'язаної з пристроєм користувача, від апаратного забезпечення до операційної системи, браузера та його конфігурацій. Фінгерпринтинг в браузері – це один з методів вебвідстеження, що представляє собою процес збору інформації через веббраузер для створення цифрового відбитку [9].

Метою фінгерпринтингу є створення індивідуального цифрового відбитка користувача на основі сукупної інформації про конфігурацію пристроїв і браузерів, як-от версія браузера та операційної системи, мова інтерфейсу, часовий пояс, встановлені шрифти, налаштування браузера, активні розширення та плагіни, дані про апаратне забезпечення (роздільна здатність екрана, використання акумулятора, пам'ять пристрою), параметри WebGL або Canvas тощо. Хоча кожна з цих деталей не дозволяє однозначно ідентифікувати користувача, всі разом вони створюють унікальний цифровий відбиток пристрою.

Цифрові відбитки браузера можна використовувати як засіб безпеки (наприклад, як засіб автентифікації користувача). Однак зняття цифрових відбитків також є потенційною загрозою конфіденційності та приватності користувачів в Інтернеті.

Фінгерпринтинг в бараузері поділяється на активний і пасивний [10].

При пасивному фінгерпринтингу, вебсайти збирають технічні характеристики пристрою та браузера користувача, які надаються автоматично без активної взаємодії з користувачем. Це може включати такі параметри, як версія браузера, тип операційної системи, розмір екрану, шрифти, мовні налаштування та інші характеристики, які можуть бути доступні через заголовки запитів HTTP або JavaScript API. Пасивний фінгерпринтинг не потребує додаткових дій з боку користувача і здійснюється без його відома.

Активний фінгерпринтинг вимагає активної взаємодії користувача з вебсайтом або додатком для збору додаткової інформації про його пристрій. Це може включати виконання скриптів JavaScript, запити на доступ до деяких API пристрою, вимогу до користувача ввести певну інформацію або навіть завантаження спеціального програмного забезпечення або додатків на пристрій користувача. Активний фінгерпринтинг дає можливість отримати детальнішу інформацію про пристрій і його налаштування.

Стовідсотковий захист від зняття цифрових відбитків неможливий, оскільки загалом є безліч параметрів, які можуть враховуватися при створенні цифрового відбитку і цілком раціонально, що ми можемо очікувати лише часткового захисту від фінгерпринтингу. Ресурс Cover Your Tracks [11] надає можливість перевірити свій рівень захисту в мережі Інтернет, який проводить тестування браузера і надає інформацію про рівень захищеності вебпереглядача користувача від фінгерпринтингу та вебтрекінгу. У табл. 1.1 наведені параметри, які оцінює зазначений ресурс, та їх значення.

У результатах тестування браузера сервісом Cover Your Tracks деякі показники можуть бути вказані як «1» або «0», або «true» або «false», що вказує на те, чи увімкнено налаштування. Хоча деталі кожного окремого показника можуть здаватися невеликою кількістю інформації, у поєднанні з іншими показниками вебпереглядача вони можуть унікально ідентифікувати браузер користувача. Результати вимірюються в «бітах ідентифікаційної інформації», яка є об’єднаним підсумком усіх цих показників.

Таблиця 1.1

Параметри для тестування Cover Your Tracks

Назва параметру	**Значення параметру**	**Біти ідентифікаційної інформації**
User-agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36 OPR/98.0.0.0	8.19
HTTP_ACCEPT headers	text/html, */*; q=0.01 gzip, deflate, br ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7	7.07
Time zone offset	-180	4.37
Time zone	Europe/Kiev	7.68
Screen size and color depth	1536x864x24	4.25
Browser plugin details	**Plugin 0:** Chrome PDF Viewer; Portable Document Format; internal-pdf-viewer; (Portable Document Format; application/pdf; pdf) **Plugin 1:** Chromium PDF Viewer; Portable Document Format; internal-pdf-viewer; (Portable Document Format; application/pdf; pdf)	0.87
System fonts	Arial, Arial Black, Arial Narrow, Book Antiqua, Bookman Old Style, Calibri, Cambria, Cambria Math, Century, Century Gothic, Century Schoolbook etc.	3.91
Are cookies enabled?	Yes	0.16
Hash of canvas fingerprint	564d9a2725ffc026efdc563c65fd2d8c	7.07
Hash of WebGL fingerprint	3099720f0a8a1481492cf0744385f14e	7.09
DNT header enabled?	False	0.86
Language	ua-UA	6.03
Platform	Win32	1.35
Audiocontext fingerprint	124.04347527516074	2.65
CPU class	N/A	0.16
Device memory (GB)	8	2.14

Однією з головних переваг використання фінгерпринтингу є відсутність збереження стану, внаслідок чого цифровий відбиток може залишатися стабільним під час кількох сеансів, серфінгу в режимі анонімного перегляду, видалення або повторного встановлення програм або очищення файлів cookie.

Кінцеві користувачі мають певний контроль над відстеженням за файлами cookie, оскільки користувач у змозі видалити або відхилити їх на певному вебсайті або в загальних налаштуваннях браузера, і в подальшому вебсайт більше не зможе розпізнавати або відстежувати діяльність користувача. У випадку з цифровими відбитками, коли браузер надсилає запит на вебсервери, різні сайти можуть об'єднувати інформацію про одного користувача і, як результат, дані про користувача збираються та групуються в унікальний профіль [12]. Завдяки цьому профілю, користувачі можуть відстежуватися та бути ідентифікованими у всьому Інтернеті.

Процес фінгерпринтингу включає збір і аналіз різноманітних технічних характеристик пристрою та браузера, до яких відносять:

- Фінгерпринтинг Canvas – це техніка, яка використовується для відстеження та ідентифікації вебкористувачів за допомогою елемента Canvas HTML5 у веббраузері користувача. HTML5 Canvas — це API із вбудованими об'єктами, методами та властивостями для зображення текстів і графіки. Браузери можуть використовувати його функції для відтворення вмісту вебсайту. Процес створення цифрового відбитку Canvas включає в себе використання API Canvas для малювання специфічних зображень та отримання даних про рендеринг цих зображень. Отримані зображення та текст відрізняються залежно від графічних можливостей пристрою. У дослідженні [13] зазначено, що параметри, такі як розмір canvas, кольори, шрифти, піксельні дані та інші технічні деталі використовуються для створення унікального відбитку. Цей відбиток може бути відправлений на сервер для подальшого використання в цілях відстеження або аналізу.

- Фінгерпринтинг WebGL— це графічний API, призначений спеціально для відтворення 3D-інтерактивних зображень. Процес створення фінгерпринта WebGL включає в себе отримання інформації про можливості графічної підсистеми браузера, такі як підтримувані розширення, типи шейдерів, максимальні розміри текстур, підтримувані формати зображень та інші параметри. Ця інформація може бути зібрана через API WebGL та використана для створення унікального відбитку [13, 14].

- Аудіофінгепринтинг (AudioContext). Основна ідея аудіофінгерпринтингу полягає у створенні унікального відбитку на основі властивостей аудіо-сигналів, які

можуть бути зібрані з мікрофону користувача у браузері. Web Audio API — це інтерфейс для обробки звуку, що використовується для отримання доступу до аудіо-потоку та обробки аудіо-сигналів. Різні атрибути аудіо-сигналів, такі як амплітуда, частота, спектральні характеристики і інші, можуть бути використані для створення унікального цифрового відбитку [15].

- Розширення браузера – це невеликі програмні модулі, які розширюють функціональні можливості та можливості браузера. Комбінацію розширень або плагінів, унікальних для браузера, можна додати безпосередньо до цифрового відбитку. Розширення також можуть змінювати поведінку будь-яких інших атрибутів браузера, додаючи додаткової складності цифровому відбитку користувача [9].

- Заголовки HTTP. Браузери надають свою назву та версію разом із деякою інформацією про сумісність у заголовку запиту User-Agent [16]. User-Agent — це рядок тексту, який надсилається кожному вебсайту, який відвідує користувач, і повідомляє, яка ОС і браузер використовується, а також їхні версії. Хоча ця інформація може бути використана, щоб показати релевантну інформацію для девайсу користувача, вона також може бути використана для його ідентифікації або зловмисниками, щоб визначити, чи його система або браузер вразливі до певних атак [9]. Тип і версію вебпереглядача також можна визначити на основі спостережень за особливостями його поведінки, зокрема порядок і кількість полів заголовків HTTP є унікальними для кожного сімейства браузерів, кожне сімейство та версія браузера відрізняється реалізацією HTML5, CSS та JavaScript [17].

- CSS запити. Різні CSS властивості, такі як ширина та висота елементів, розміри шрифтів, колір тексту та фону, відстані між елементами, орієнтація екрана, співвідношення сторін дисплея тощо, можуть мати варіації між різними браузерами та конфігураціями, що призводить до унікальних характеристик CSS стилів. Ці варіації можуть використовуватись для створення унікального відбитка браузера, який може бути використаний для відстежування користувача в межах різних вебсайтів і сеансів перегляду [13].

- Встановлені шрифти. Ідентифікація користувачів на основі відтворених текстів також є одним з параметрів фінгерпринтингу, оскільки браузери на різних

пристроях відтворюють однаковий стиль шрифту та символи з різними обмежувальними рамками. Отже, сайти можуть запитувати обмежувальні рамки для збору даних цифрових відбитків [14].

- Порівняльний аналіз обладнання можна використовувати, наприклад для порівняння часу центрального процесору, який використовується для виконання різних простих або криптографічних алгоритмів. Також можна використовувати спеціалізовані API, такі як Battery API [9], який створює короткочасний відбиток пальця на основі фактичного стану батареї пристрою або OscillatorNode, який можна викликати для створення хвилі на основі ентропії користувача. Ідентифікатор апаратного забезпечення пристрою, який є криптографічною хеш-функцією, визначеною постачальником пристрою, також можна запитувати для створення відбитка.

- Сканування історії вебперегляду. Сканер цифрових відбитків може визначити, які сайти вебпереглядач раніше відвідував, запитуючи список за допомогою JavaScript із селектором CSS :visited.

4. Піксель відстеження або вебмаяк — це файл зображення (зазвичай прозорий GIF), який містить код, який може слідкувати, чи відкривав користувач електронний лист або відвідував вебсайт. Код вбудовано в зображення, і користувач не бачить його. Коли він відкриває електронний лист або відвідує вебсайт, який містить пікселі відстеження, код виконується та відстежує його діяльність. Потім ці дані можуть бути використані маркетологами для аналізу поведінки відвідувачів і відповідної оптимізації їхніх маркетингових стратегій [18].

Оператори вебсайтів можуть вставляти різні «пікселі» (маленькі фрагменти коду) різних служб свій вебсайт. Популярними плагінами вебтрекінгу, які використовують піксель відстеження є Facebook Pixel, Google Analytics, PixelYourSite тощо. Виконуючи певну дію відвідувача (наприклад, прокручування), піксель збирає відповідні дані та надсилає їх сервісу. Дія для відстеження може бути визначена оператором сайту під час налаштування пікселя (так звана «подія»). В деяких випадках компанії використовують файли cookie спільно з вебмаяком. Якщо власник

пікселя також зберіг файл cookie на пристрої користувача, його дії на вебсайті можна пов'язати з усім у профілі, який компанія вже створила для нього.

1.4 Аналіз існуючих рішень для забезпечення анонімності

1.4.1 Інструменти для приховування IP-адреси

До інструментів, що використовуються для приховування IP-адреси відносять:

1. Проксі-сервер - це серверна програма, що діє як проміжний сервер, який дозволяє користувачеві інтернету зберігати свою анонімність та безпеку в мережі Інтернет, приховуючи його IP-адресу від серверів вебсайтів, які він відвідує. Зазвичай, проксі-сервери використовуються з метою забезпечення анонімності користувача, захисту його приватної інформації та обходу блокування вебсайтів [19]. Алгоритм роботи проксі-сервера полягає у тому, що коли користувач запитує вебсторінку, він не звертається безпосередньо до сервера, який містить цю сторінку. Замість цього, він звертається до проксі-сервера, який відправляє запит на вебсторінку від імені користувача. Після цього проксі-сервер передає відповідь від сервера користувачеві (рис. 1.1).

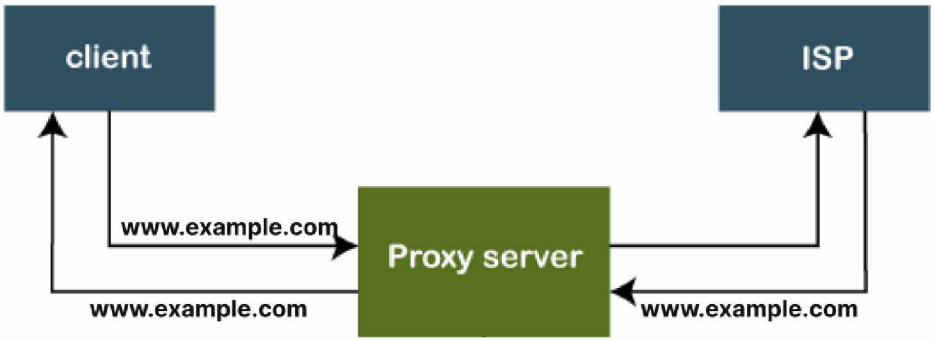

Рисунок 1.1 – Механізм роботи проксі-серверів

Загалом є багато типів проксі-серверів, але не всі проксі є анонімними, і рівень анонімності залежить від їх типу. Проксі змінює дані заголовка на різних рівнях, і на основі рівня анонімності проксі поділяються на три категорії: прозорі, анонімні та елітні [20].

Прозорі проксі не створюють захист анонімності та показують IP-адресу користувача і не приховують факт підключення через проксі. Прозорі проксі використовуються в публічних мережах Wi-Fi і стежать за поведінкою підключених користувачів у вебпереглядачі та можуть обмежувати доступ до деяких вебсайтів. З'єднання через прозорий проксі є незахищеним, оскільки воно не зашифроване. Прикладами прозорих проксі-серверів є Squid, Polipo, Apache Traffic Server, HAProxy.

Анонімні проксі, на відміну від прозорих, не показують IP-адресу користувача. Він замінює IP-адресу на нову, але залишає HTTP_VIA в заголовку, інформуючи цільовий сайт, що відвідувач підключається через проксі. Навіть якщо IP-адреса прихована, деякі вебсайти можуть в такому випадку обмежити доступ до вмісту. ПЗ, яке надає доступ до анонімних проксі - Proxy Server, ProxySite.com, Freegate.

Елітні (високоанонімні) проксі-сервери – найбільш оптимальний варіант для підвищення рівня онлайн-анонімності. Вони приховують будь-яку інформацію, яка може обмежити доступ користувача до певного сайту. IP-адреса не лише видаляється із заголовка запиту, але й саме з'єднання стає ідентичним фактичному запиту пристрою. ПЗ, що надає доступ до високоанонімних проксі-серверів - Smartproxy, ProxyBonanza, Luminati тощо.

Внаслідок того, що анонімні проксі приховують IP-адресу користувача, з'являється можливість уникати цільового маркетингу та реклами в Інтернеті, обходити геоблокування та уникати цензури, покращити свою безпеку та запобігти крадіжці особистих даних, захистити історію пошуку.

Проте проксі-сервери також мають значні недоліки. По-перше, сама технологія обмежена: проксі-сервери є вузькоспеціалізованими, тому для кожного типу підключення до Інтернету потрібен окремий тип проксі. По-друге, всі типи проксі мають важливу спільну проблему безпеки, оскільки проксі-сервери додатково жодним чином не шифрують трафік, тому кожен проміжний сервер потенційно може ознайомитись з інформацією, яка передається в запитах на цільовий вебсайт.

2. VPN (Virtual Private Network) - це технологія, що дозволяє створювати приватну мережу через публічну мережу Інтернет. В основі роботи VPN лежить процес шифрування трафіку, який забезпечує конфіденційність даних, що

передаються між комп'ютерами [21]. Принцип роботи VPN полягає в тому, що користувач підключається до сервера VPN-провайдера, який стає посередником між користувачем і Інтернетом. У цьому випадку, всі дані, які надходять з пристрою користувача, шифруються та передаються через захищенний тунель до сервера VPN-провайдера, де вони розшифровуються та надсилаються до необхідного сайту чи сервісу (рис. 1.2).

Механізм роботи технології VPN можна описати наступним чином:

- Встановлення з'єднання: Користувач запускає VPN-клієнт або налаштовує VPN-підключення на своєму пристрої. Клієнтська програма VPN встановлює з'єднання з сервером VPN провайдера.
- Шифрування даних: коли з'єднання встановлено, усі дані, що передаються між користувачем і сервером VPN, шифруються за допомогою різних шифрувальних протоколів, таких як OpenVPN, IPsec, PPTP або L2TP/IPsec. Шифрування даних забезпечує конфіденційність та захист від перехоплення чи прослуховування.
- Заміна IP-адреси: коли дані відправляються через VPN-з'єднання, IP-адреса користувача заміняється IP-адресою сервера VPN. Це приховує реальну IP-адресу користувача і робить його анонімним у мережі.
- Маршрутизація трафіку: після шифрування і заміни IP-адреси дані направляються через зашифрований тунель між клієнтом і сервером VPN. Сервер VPN, виконуючи роль посередника, пересилає зашифровані дані до вебсайтів або інших ресурсів, до яких користувач хоче отримати доступ.
- Дешифрування на сервері VPN: на сервері VPN зашифровані дані розшифровуються і надсилаються до призначеного вебсайту або ресурсу. Сервер VPN діє в якості посередника між користувачем і цільовим ресурсом.
- Отримання відповіді: після отримання відповіді від цільового ресурсу сервер VPN шифрує дані, передає їх через зашифрований тунель до VPN-клієнта і далі розшифровує на пристрої користувача.

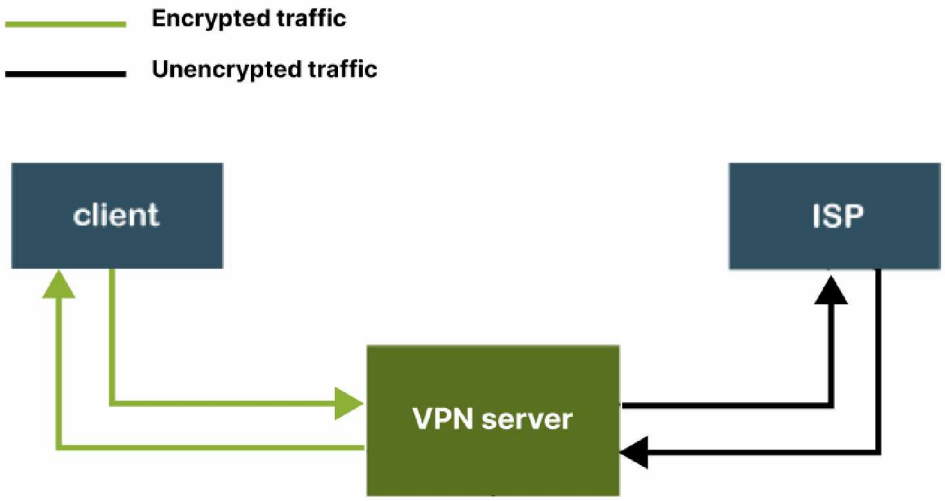

Рисунок 1.2 – Механізм роботи VPN

Таким чином, VPN-технологія дозволяє забезпечити конфіденційність даних, що передаються між комп'ютерами, а також змінювати IP-адресу користувача для забезпечення анонімності в мережі Інтернет. Крім того, використання VPN-сервера дозволяє заблокувати доступ до інформації користувача з боку провайдера Інтернет-послуг. Проте VPN не є панацеєю для забезпечення анонімності, тому що деякі додатки та сервіси можуть виявляти користувача за його поведінкою в Інтернеті, або за іншими параметрами, які не можна змінити через VPN.

VPN безпечніші за проксі, оскільки вони використовують розширені алгоритми шифрування, такі як AES-256 і ChaCha20, для шифрування з'єднання та анонімізації трафіку. Вибір VPN-провайдера має важливе значення, тому що не всі з них забезпечують високий рівень захисту користувача.

Існують як безкоштовні рішення, так і платні, однак очевидно, що безкоштовні сервіси мають ряд недоліків, такі як невисокий рівень найдійності, обмежений функціонал, низька швидкість тощо. Як і у випадку з проксі-серверами, безкоштовні VPN-сервіси неодноразово виявлялися у шпигунстві за користувачами та продажу їхніх даних[22]. Проте, якщо бюджет обмежений, то з безкоштовних (або наявних безкоштовних версій) можна використовувати, наприклад, ExpressVPN, CyberGhost, Proton VPN, Avira Phantom VPN тощо. З платних VPN сервісів – NordVPN, Surfshark VPN, AtlasVPN, ExpressVPN тощо.

3. TOR (The Onion Router) - це безкоштовний відкритий інструмент для забезпечення анонімності в Інтернеті, який дозволяє користувачам приховати свою Інтернет-діяльність від сторонніх осіб, включаючи провайдерів Інтернету та державних органів [21, 23, 24]. Основний принцип роботи TOR полягає у використанні мережі вузлів, які шифрують трафік між користувачем та кінцевим сайтом, роблячи його недоступним для прослуховування. Щоб анонімізувати Tor, як і проксі та VPN, він пропускає трафік через проміжні сервери. Але тільки у випадку з TOR їх не один, а три, і вони називаються вузлами. Для створення мережі TOR користувачі на своїх комп'ютерах розгортають ці вузли: чим більше користувачів, тим безпечніша та швидша мережа.

Трафік проходить через три вузли: вхідний, проміжний, вихідний. Кожен вузол TOR знає лише про попередній та наступний вузол, з якими він зв'язується (рис. 1.3).

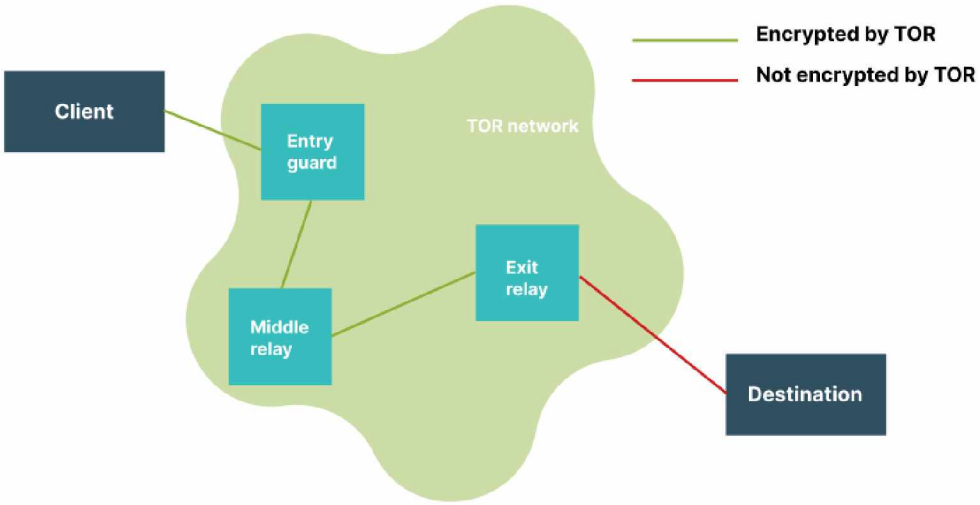

Рисунок 1.3 – Механізм роботи мережі TOR

Коли користувач намагається отримати доступ до вебсайту через TOR, його запит переадресовується через випадковий шлях вузлів у мережі, кожен з яких шифрує трафік знову та знову, додавши до запиту додатковий шар шифрування. Коли запит досягає останнього вузла в мережі, він дешифрується та передається до кінцевого вебсайту. І оскільки кожен вузол знає IP-адресу лише попереднього вузла в ланцюжку, то вихідний IP буде втрачено, коли трафік користувача досягне

останнього вузла. Цей процес забезпечує високий рівень анонімності, оскільки неможливо відслідкувати весь маршрут, який зробив запит у мережі TOR.

Проте TOR має і ряд недоліків, серед яких, по-перше, досить повільна швидкість з'єднання через багатошаровий процес шифрування трафіку – це служить причиною того, що багато сайтів некоректно працюють через TOR браузер. По-друге, факт використання TOR легко відстежити, з чого випливає, що через TOR боаузер користувач потенційно може привернути небажану увагу правоохоронних органів, тому що TOR часто використовується в тому числі і для незаконних дій. По-третє, власники вихідних вузлів дуже ризикують, адже саме вони відповідають за всі дії, які здійснюють користувачі мережі TOR. Власники вихідних вузлів бачать увесь трафік користувача, тому опосередковано можуть відстежувати його.

Використання «ланцюгів анонімності». Комбінування описаних вище технологій дозволяє підвищити рівень анонімності користувача в мережі. Існує декілька способів поєднання технологій TOR, VPN і Proxy у так звані «ланцюги анонімності» для забезпечення більшої анонімності в Інтернеті [25], зокрема:

- VPN + TOR: користувачі можуть спочатку підключитися до VPN-сервера, а потім до мережі TOR. Це дозволить захистити користувача від можливих витоків DNS, а також додатково зашифрувати трафік, який вже шифрується в мережі TOR.
- TOR + VPN: користувачі можуть спочатку підключитися до мережі TOR, а потім до VPN-сервера. Це дозволить приховати від VPN-провайдера те, що користувач використовує VPN, і зменшити ризик витоків DNS.
- Proxy + TOR: користувачі використовують проксі-сервери для забезпечення додаткового шару анонімності і зменшення ризику витоків інформації.
- TOR + VPN + Proxy: цей метод дозволяє захистити IP-адресу користувача від проксі-сервера і шифрувати весь інтернет-трафік між користувачем та VPN-сервером. Крім того, TOR дозволяє приховати користувача від VPN-провайдера.
- VPN + TOR + Proxy: забезпечує захист від DNS-витоків і дозволяє користувачеві отримати доступ до змісту, який може бути заблокований у деяких країнах або регіонах.

1.4.2 Вбудовані функції безпеки в браузерах

В даному пункті будуть розглянуті браузери Mozilla Firefox, Apple Safari та Google Chrome та виконане порівняння їх функціоналу, наявного для захисту від вебтрекінгу.

Mozilla Firefox — це браузер із відкритим кодом, створений некомерційною організацією Mozilla. Даний вебпереглядач зосереджується на захисті приватності та безпеці користувача, надаючи різноманітні інструменти і функції [26].

Mozilla Firefox пропонує низку вбудованих функцій для захисту від вебстеження та забезпечення анонімності користувача, такі як:

1. Enhanced Tracking Protection: функція покращеного захисту від відстеження Firefox блокує сторонні файли cookie, які часто використовуються рекламодавцями для відстеження користувачів на кількох вебсайтах. Ця функція ввімкнена за замовчуванням і може бути налаштована для відстеження на певних сайтах [27].

У Mozilla Firefox доступні три режими для ETP: стандартний, строгий і спеціальний [28]:

- Стандартний режим блокує відомі трекери в приватних вікнах і файли cookie сторонніх розробників, криптомайнери та сканери цифрових відбитків. Однак він все ще дозволяє деякі трекери, які Firefox вважає безпечними.
- Строгий режим забезпечує покращений захист конфіденційності шляхом блокування відомих трекерів у всіх вікнах, а не лише в приватних вікнах. Він також блокує всі файли cookie сторонніх розробників, криптомайнери та сканери цифрових відбитків. Цей режим іноді може зламати певні вебсайти, які залежать від цих трекерів, але він забезпечує найвищий рівень захисту конфіденційності.
- Спеціальний режим дозволяє користувачеві обирати, які трекери блокувати, а які дозволяти. Користувачі можуть заблокувати сторонні файли cookie, криптомайнери та відбитки пальців, а також вибрати зі списку відомих трекерів для блокування.

2. First-Party Isolation: розділяє файли cookie та інші дані, що зберігаються вебсайтами, у різні контейнери, запобігаючи відстеженню користувачів на різних сайтах.

3. Privacy Settings: Firefox дозволяє користувачам налаштовувати свої параметри конфіденційності, щоб контролювати, скільки даних вебсайти можуть збирати про них. Користувачі можуть заблокувати всі файли cookie, блокувати сторонні файли cookie і видалити файли cookie та дані сайту, коли Firefox закрито.

4. Private Browsing: вікно приватного перегляду в Firefox захищає дані користувача від фінгерпринтингу і відстеження та за замовчуванням видаляє сторонні файли cookie та трекери разом з історією вебперегляду, коли користувач закриває вкладку.

5. HTTPS-Only Mode: дозволяє користувачам переглядати лише вебсайти, які використовують шифрування HTTPS, забезпечуючи додатковий рівень безпеки та конфіденційності.

6. Do Not Track: Firefox підтримує стандарт «Не відстежувати» (DNT), який повідомляє вебсайтам, що користувач не хоче, щоб його відстежували. Однак, вебсайти не завжди дотримуються цього побажання своїх відвідувачів.

7. Fingerprint protection: функція, яка призначена для запобігання вебсайтам відстеження користувачів на основі унікальних характеристик браузера, таких як роздільна здатність екрана, встановлені шрифти та плагіни браузера. Ця функція доступна в налаштуваннях конфіденційності.

Firefox борються з Canvas фінгерпринтингом за допомогою списків фільтрів, які є реєстрами доменів, що виконують дії, які фахівці Firefox розцінюють як порушення конфіденційності. Firefox шукають скрипти компаній, які застосовують Canvas фінгерпринтинг, і відмовляються надсилати цим доменам будь-яку інформацію про користувачів.

Apple Safari – це веббраузер, розроблений компанією Apple, який має вбудовані функції безпеки та захисту приватності користувача [29], до яких відносять наступні:

1. Intelligent Tracking Prevention (ITP): дозволяє Safari блокувати трекери сторонніх ресурсів, що збирають дані про користувача, його активність та іншу

інформацію про перегляд вебсайтів. ITP аналізує поведінку користувачів та автоматично блокує спроби вебтрекінгу. Він блокує міжсайтові файли cookie за допомогою алгоритму машинного навчання та приховує цифрові відбитки, щоб вебсайти не залишали трекери та скрипти у вашому сховищі. ITP також обмежує використання основних файлів cookie для міжсайтового відстеження [30].

2. Privacy Report: панель, що надає користувачу зведену інформацію про сайти, які намагалися відстежувати його активність. Вона також надає користувачу інформацію про те, які файли cookie були збережені на його пристрої. Safari пропонує звіт про конфіденційність, який показує користувачу кількість реклами та трекерів, які були заблоковані.

3. Intelligent Anti-Fingerprinting: дозволяє запобігти створенню унікального ідентифікатора пристрою шляхом збільшення розміру вікна браузера та додавання випадкових елементів до параметрів пристрою.

4. Device fingerprinting prevention: Safari обмежує кількість інформації, до якої можуть отримати доступ вебсайти, щоб створити унікальний відбиток пальця пристрою. Це включає обмеження доступу до деталей конфігурації обладнання та встановлених шрифтів. Safari бореться з відбитками пальців, намагаючись зробити пристрій кожного користувача максимально схожим, і оскільки більшість користувачів Safari використовують пристрої Apple, це полегшує задачу.

5. Private Browsing: режим, що дозволяє користувачеві переглядати вебсторінки без збереження історії перегляду та файлів cookie на пристрої. DuckDuckGo встановлена як пошукова система за замовчуванням у приватному вікні.

6. Password Monitoring: функція, що перевіряє, чи були паролі, збережені у Safari, викрадені в результаті порушення безпеки даних.

Google Chrome – найпопулярніший у світі браузер, яким користується близько 70 відсотків користувачів, розроблений компанією Google на основі браузера з відкритим кодом Chromium [26].

Вбудовані функції Google Chrome для захисту від відстеження в Інтернеті та забезпечення анонімності користувачів включають в себе [31]:

1. Incognito Mode: режим анонімного перегляду розроблений для конфіденційного перегляду вебсторінкок. Коли користувач відкриває вікно в режимі анонімного перегляду, Chrome не зберігає історію вебперегляду, файли cookie, дані сайтів або інформацію, яку користувач вводить у формах.

2. Блокування файлів cookie: Chrome за замовчуванням дозволяє сторонні файли cookie, але вони автоматично блокуються у режимі анонімного перегляду. Користувачі Chrome, які хочуть заблокувати сторонні файли cookie можуть змінити налаштування конфіденційності вебпереглядача. Chrome надає користувачам можливість блокувати всі основні файли cookie в налаштуваннях конфіденційності.

3. Safe Browsing: функція безпечного перегляду попереджає користувача перед відвідуванням небезпечних вебсайтів. Ця функція допомагає захистити користувача від зловмисного програмного забезпечення, фішингу тощо.

4. Налаштування сайту: налаштування сайту Chrome дозволяють контролювати, як вебсайти взаємодіють із пристроєм користувача (є можливість контролювати, чи мають вебсайти доступ до місцезнаходження, камери та мікрофона відвідувача вебсайту).

5. DNS-over-HTTPS: Chrome підтримує DNS-over-HTTPS (DoH), що допомагає захистити DNS-запити користувача від перехоплення та відстеження.

6. Підтримка Do Not Track: Chrome підтримує заголовок Do Not Track, який можна включити в налаштуваннях браузера.

7. Блокування вмісту: Chrome має вбудований блокувальник реклами та блокувальник шкідливих вебсайтів.

Порівняння Firefox, Chrome та Safari. Дослідження [32] виявило велику різницю між тим, що рекламодавці знали про людей, які використовують Safari, який за умовчанням блокує файли cookie, і Chrome, який цього не робить. За результатами, 73 відсотки оголошень, показаних користувачам Safari, не змогли пов’язати людину з рекламним профілем, ймовірно, через це блокування, тоді як лише 17 відсотків реклами, показаної користувачам браузера Google Chrome, не змогли пов’язати користувача з профілем. Детальніше порівняння вбудованих функцій захисту розглянутих вебпереглядачів наведене у табл. 1.2.

Таблиця 1.2

Порівняння функціоналу браузерів за захистом від вебтрекінгу

Функція	**Mozilla Firefox**	**Google Chrome**	**Apple Safari**
Блокування сторонніх cookies	Доступно (за замовчуванням)	Доступно (за додаткових налаштувань)	Доступно (за замовчуванням)
Вбудований захист від вебтрекінгу	Доступно (за замовчуванням)	Частково (тільки для сторонніх файлів cookies)	Доступно (за замовчуванням)
Автоматичне видалення основних файлів cookie	Доступно (за замовчуванням, якщо не було взаємодії з вебсайтом 45 днів)	Доступно (за додаткових налаштувань)	Доступно (за замовчуванням через 7 днів)
Блокування спроб фінгерпринтингу	Доступно (за замовчуванням)	Немає	Доступно (за замовчуванням)
?Блокування реклами	Доступно (за замовчуванням, за допомогою розширень)	Доступно	Доступно (за замовчуванням)
Do Not Track	Доступно	Доступно	Недоступно
Відключення Javascript	Доступно	Доступно	Доступно
Режим приватного перегляду	Доступно	Доступно	Доступно
Попередження про відстеження	Доступно	Немає	Доступно
DNS-over-HTTPS	Доступно	Доступно	Доступно
Використання пошуку без збору даних	Доступно (за допомогою вбудованого пошукового двигуна DuckDuckGo)	Недоступно	Доступно (за допомогою вбудованого пошукового двигуна DuckDuckGo)
Захист від кейлогінгу	Доступно	Немає	Доступно
Захист від запису сеансу	Доступно	Немає	Немає

Усі три браузери, Firefox, Chrome та Safari, мають свої засоби для захисту конфіденційності користувачів. Однак, як видно з табл. 1.2, Firefox та Safari мають більш широкий функціонал, що дозволяє краще захищати користувача від вебтрекінгу.

Firefox має вбудовані інструменти для блокування трекерів та захисту від зчитування цифрових відбитків, а також підтримує Do Not Track функцію, яка дозволяє надсилати запити до вебсайтів про відмову від відстеження користувача. Також, Firefox має можливість блокування сторонніх cookies за замовчуванням та автоматичне видалення основних файлів cookie.

Safari також має вбудовані інструменти для блокування трекерів та захисту від зчитування цифрових відбитків, а також підтримує функцію Intelligent Tracking Prevention, яка дозволяє блокувати трекери та перешкоджає їхній роботі. Safari також має функцію, що дозволяє блокувати сторонні cookies та автоматичне видалення файлів cookie.

Chrome також має засоби для захисту конфіденційності, такі як можливість блокування трекерів та реклами, однак Chrome має менше функцій захисту від вебтрекінгу, а тому буде менш ефективний, ніж в Firefox або Safari. Chrome має функцію Safe Browsing, яка дозволяє блокувати небезпечні вебсайти та завантаження шкідливих файлів, але ця функція не пов'язана з конфіденційністю користувача.

Таким чином, Firefox та Safari пропонують сильніший захист конфіденційності порівняно з Chrome. Однак, усі три браузери дозволяють використовувати розширення для блокування трекерів, фінгепринтингу та інших небажаних елементів вебсторінок, наприклад такі як Privacy Badger, Canvasblocker, Trace, що будуть розглянуті в наступному підрозділі.

1.4.3 Програмні розширення для запобігання вебтрекінгу

Для більш надійного захисту від стеження рекомендується використовувати розширення для браузерів, що допомагають захиститися від стеження на більш високому рівні, такі як Privacy Badger, Canvasblocker, Trace. В цьому пункті будуть розглянуті детальніше кожне з перелічених розширень, його функціонал та тестування їх ефективності захисту проти вебтрекінгу та фінгерпринтингу в браузері.

Trace - це розширення для браузера з відкритим вихідним кодом, яке допомагає користувачам контролювати рівень свого захисту від вебтрекінгу та фінгерпринтингу, керувати функціями браузера, які важко знайти, або які увімкнено без можливості вимкнути. Розширення аналізує вебсторінки на наявність трекерів, cookies та інших елементів, які можуть порушити конфіденційність користувачів. Він доступний для Google Chrome, Mozilla Firefox і Microsoft Edge [33].

Розробники заявляють, що дане розширення має низку корисних функцій для захисту від фінгерпринтингу, зокрема:

1. Захист від canvas-фінгерпринтингу - Trace випадково генерує новий хеш canvas для кожного запиту, унеможливлюючи прив'язування вас до однієї особи.

2. Захист від аудіофінгерпринтингу - вимикає AudioContext API.

3. Захист від WebGL-фінгерпринтингу - запобігає від створення відбитку на основі моделі графічного процесора та кількості текстур, яку він може відтворити; рендереру WebGL, унікального на кожному пристрої шляхом WebGL API.

4. Client Rects Fingerprinting Protection - ця функція Javascript повертає інформацію про координати та розмір HTML-елементів на вебсторінці. Всі повернуті значення є пікселями, Trace додає рандомізоване десяткове зміщення до кожного значення, що дозволяє успішно підробити цифровий відбиток, не вплинувши на коректну роботу вебсайту.

5. Захист файлів cookie HTTP - Trace захищає користувача від сторонніх файлів cookie, що використовуються для відстеження, перехоплюючи заголовки Set-Cookie і Cookie, звіряючи назви файлів cookie, перевіряючи, чи є вони сторонніми чи основними, а потім видаляє їх залежно від налаштувань користувача.

6. Захист від зчитування роздільної здатності екрана - Trace може модифікувати змінні браузера, які зчитують вебсайти, що призводить до того, що користувач має новий ідентифікатор при кожному завантаженні сторінки.

7. Контроль над заголовком Referrer. HTTP-заголовок «Referer» — це заголовок, який ідентифікує для вебсайту URL-адресу сторінки, з якої надійшов запит на ресурс.

8. Видалення спеціальних заголовків Chrome - з вебзапитів видаляються заголовки X-Client-Data, X-Chrome-UMA-Enabled, X-Chrome-Variations і X-Chrome-Connected для запобігання нав'язування експериментальних функцій у браузері, які витрачають пропускну здатність.

9. JS Plugin Spoofer - Trace замінює об'єкт плагінів Javascript, таким чином імітуючи, що у користувача не встановлено жодного плагіна.

10. Зміна агента користувача - значення агента користувача HTTP змінюється на випадкове кожні 15 секунд.

11. Запобігання витоку WebRTC - за замовчуванням технологія WebRTC повідомляє вебсайту локальну IP-адресу користувача, також може повідомляти про інші пристрої у мережі [34]. Блокування WebRTC не лише ускладнює відстеження, але й може завадити зловмисникам оглянути мережу та знайти потенційні вразливості.

Коли користувач виконав інсталяцію розширення в браузері, перед ним відкривається початкове вікно, в якому користувач може обрати рівень захищеності (рис. 1.4). Trace має п’ять рівнів захисту – мінімальний, стандартний (за замовчуванням), високий, екстримальний, регульований.

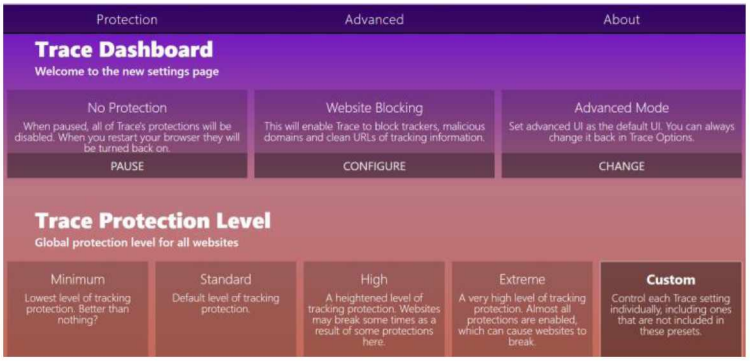

Рисунок 1.4 – Початкова сторінка Trace

Після відвідування вебсайту, переглянувши повідомлення від Trace, побачимо кількість заблокованих вебзапитів від відвіданого ресурсу (рис. 1.5).

Рисунок 1.5– Повідомлення Trace про блокування вебзапитів від сайтів

У налаштуваннях Trace є можливість переглянути статистику заблокованих запитів (рис. 1.6).

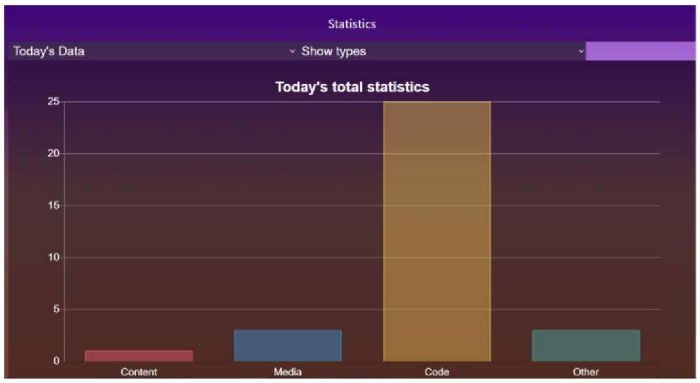

Рисунок 1.6 – Статистика заблокованих вебзапитів Trace

Для тестування використаємо сервіс Cover Your Tracks [11]. Без використання розширення Trace результати будуть наступними (рис. 1.7).

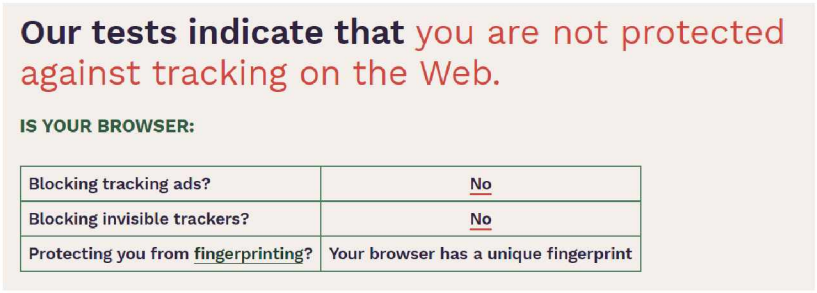

Рисунок 1.7 – Результат тестування браузера без використання розширень для захисту від вебтрекінгу

З використанням розширення Trace результати змінюються і мають вигляд, як наведено на рис. 1.8.

Рисунок 1.8 – Результат тестування браузера з використанням розширення Trace

Проаналізувавши значення параметрів, розуміємо, що були модифіковані деякі значення таким чином, як показано в табл. 1.3.

Таблиця 1.3

Значення, що були змінені за допомогою розширення Trace

Параметр	Модифіковане значення
BROWSER PLUGIN DETAILS	randomized by first party domain
HASH OF CANVAS FINGERPRINT	randomized by first party domain
HASH OF WEBGL FINGERPRINT	randomized by first party domain
WEBGL VENDOR & RENDERER	Google Inc.~Intel(R) HD Graphics
HARDWARE CONCURRENCY	randomized
DEVICE MEMORY (GB)	4

Оглянувши функціонал і протестувавши інструмент Trace, можемо прийти до висновку, що згідно з результатами сканування браузера, дане розширення цілком ефективне для захисту від фінгерпринтигу, оскільки воно успішно виконує рандомізацію цифрового відбитку користувача. Проте, Trace не надає захист від сторонніх трекерів.

Як вже відомо з попередніх пунктів, елемент canvas може використовуватися вебсайтами для відстеження поведінки користувачів, збору інформації про пристрої та зняття цифрових відбитків, які можна використовувати для ідентифікації користувача на різних вебсайтах. Canvas Blocker – розширення для вебпереглядача, призначене для захисту конфіденційних даних користувачів від вебсайтів, які використовують технологію canvas фінгерпринтинг для відстеження дій користувачів [35]. Принцип роботи Canvas Blocker полягає у блокуванні певних API, які використовуються вебсайтами для доступу до елемента canvas, а також у зміні характеристик елемента canvas, щоб перешкоджати вебсайтам збирати дані про браузер користувача. Це включає зміну розмірів і властивостей елемента canvas, а також зміну способу відтворення шрифтів і зображень. Canvas Blocker доступний як розширення браузера для Firefox і Chrome і може бути налаштований таким чином, щоб певні вебсайти отримували доступ до елемента canvas, блокуючи інші.

Основні функції розширення Canvas Blocker:

1. Захист від Canvas фінгерпринтингу: розширення дозволяє заблокувати використання Canvas API, що запобігає створенню унікальних відбитків пальців браузера і зменшує ймовірність відстеження користувача.

2. Імітація відбитку пальців: дозволяє налаштувати певні параметри Canvas API для імітації відбитку пальців, що зменшує можливість відстеження користувача.

3. Захист від WebRTC: блокує доступ до WebRTC API, що дозволяє вебсайтам отримувати IP-адресу користувача.

4. Відключення анімації: дозволяє вимкнути всі анімаційні ефекти на вебсторінках, що може зробити перегляд вебсторінок менш виснажливим для очей.

5. Блокування сторонніх скриптів: блокує завантаження сторонніх скриптів на вебсторінках, що зменшує ймовірність відстеження користувача.

Після встановлення Canvas Blocker, розширення буде працювати у фоновому режимі в браузері, постійно захищаючи користувача від canvas фінгерпринтингу. Коли користувач відвідуватиме певний вебсайт, який виконає спробу canvas фінгерпринтингу, від розширення надходитимуть повідомлення про блокування цих спроб (рис. 1.9).

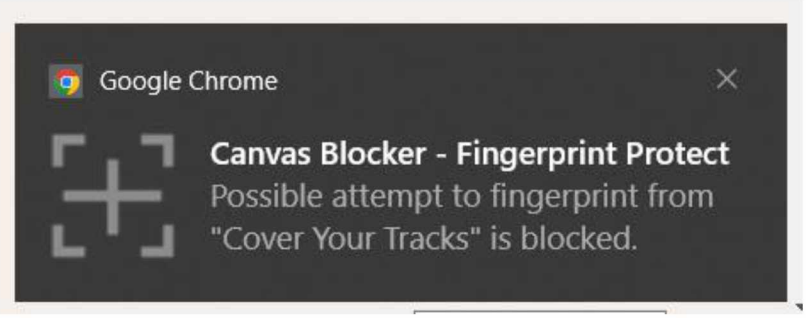

Рисунок 1.9 – Успішне блокування спроби canvas фінгерпринтингу

Виконавши тестування із активним розширенням Canvasblocker, результати виходять ідентичні, як і без його використання (рис. 1.10).

Рисунок 1.10 – Тестування розширення Canvas Blocker

Проаналізувавши значення всіх параметрів, які були перевірені, бачимо, що був модифікований тільки один параметр – значення відбитку Canvas (рис. 1.11).

Рисунок 1.11– Модифіковане значення параметру «Hash Of Canvas Fingerprint»

Отже, виконавши огляд інструменту Canvas Blocker приходимо до висновку, що він успішно виконує заявлену функцію – рандомізує значення хешу Canvas для цифрового відбитку. Проте як показує результат тестування, підміни тільки даного одного параметру недостатньо для належного захисту від фінгерпринтингу. Стосовно запобіганню вебтрекінгу та блокування відповідних програм, дане розширення не має функціоналу, щоб задовольнити дану потребу.

Privacy Badger — це безкоштовне розширення для браузера з відкритим вихідним кодом, розроблене Electronic Frontier Foundation, яке допомагає захистити конфіденційність користувачів, блокуючи онлайн-трекери. Він доступний для таких веббраузерів, як Chrome, Firefox, Opera та Edge. Його основна мета — захист конфіденційності користувачів шляхом блокування сторонніх трекерів, які збирають інформацію про їхню поведінку в Інтернеті без їхньої згоди.

Privacy Badger використовує алгоритми для ідентифікації та блокування потенційно небезпечних доменів третіх сторін, які можуть використовуватися для відстеження, профілювання або показу реклами. Розширення створює список цих доменів і пов’язаних з ними файлів cookie, які потім автоматично блокуються, коли користувачі відвідують вебсайт, який їх містить [36].

Privacy Badger використовує алгоритми машинного навчання, щоб ідентифікувати та блокувати трекери на основі їх поведінки, щойно вони з’являються,

а не покладається на заздалегідь визначені чорні списки трекерів або рекламодавців, які можуть швидко застаріти або стати неповними. Розширення також надає користувачам чіткий візуальний індикатор того, скільки трекерів було заблоковано на вебсайті, надаючи їм більше прозорості та контролю над своєю конфіденційністю в Інтернеті.

Privacy Badger також має функцію під назвою «Privacy Badger learns», яка дозволяє користувачеві вручну вказати розширенню блокувати або розблокувати певні трекери для кожного сайту. Privacy Badger може виявляти цифрові відбитки на основі canvas і блокувати сторонні домени, які використовують дану техніку фінгерпринтингу.

Після встановлення Privacy Badger працює у фоновому режимі, аналізуючи поведінку сторонніх трекерів на вебсайтах, які відвідує користувач, та блокує виявлені трекери, які відстежують поведінку користувача без його згоди (рис. 1.12).

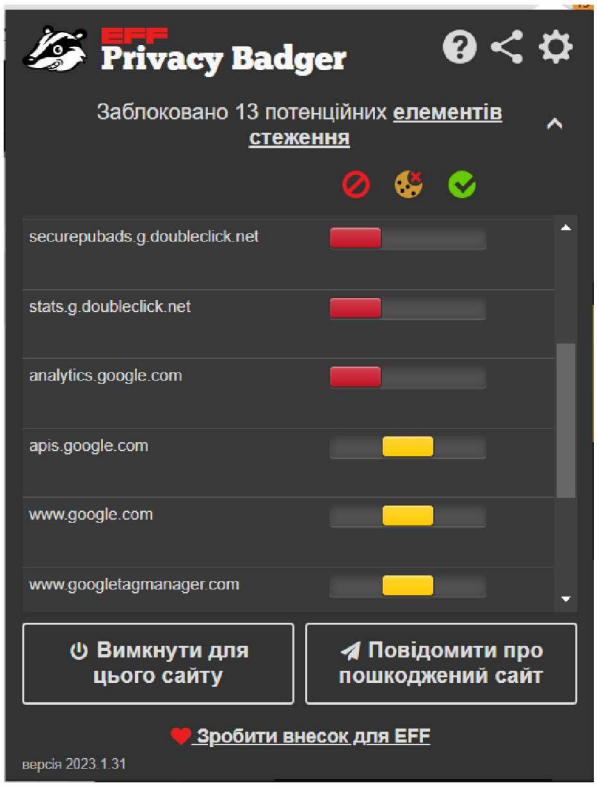

Рисунок 1.12– Список заблокованих трекерів Privacy Badger

Виконавши тестування Privacy Badger, отримуємо результат, зображений на рис. 1.13.

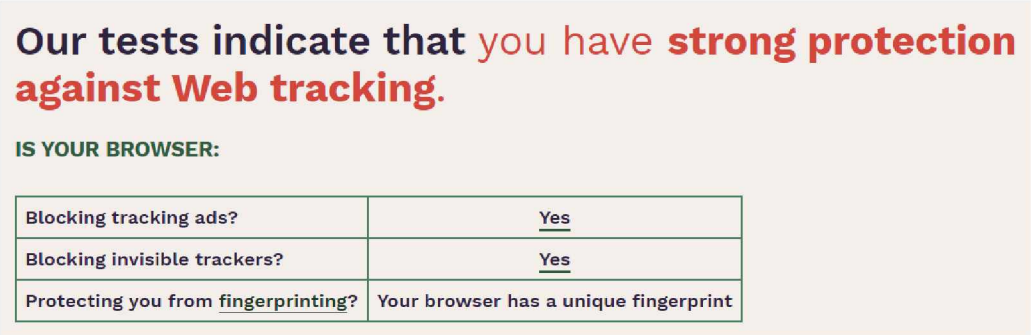

Рисунок 1.13 – Результат тестування Privacy Badger

Як бачимо з результатів тестування, дане розширення успішно виконує заявлені розробником функції – блокує трекери та рекламу, натомість не рандомізує цифровий відбиток браузера користувача. Загалом, Privacy Badger — це потужний інструмент, який може допомогти користувачам контролювати свою конфіденційність в Інтернеті та обмежити кількість даних, зібраних про них сторонніми трекерами. Проте не надає належного захисту від фінгерпринтингу в браузері.

Для узагальнення здійснимо порівняння функціоналу згаданих вище інструментів у табл. 1.4.

Таблиця 1.4

Функціонал Privacy Badger, Canvasblocker, Trace

Функція	Trace	Canvas Blocker	Privacy Badger
Захист від canvas-фінгерпринтингу	+	+	+
Захист від WebGL-фінгерпринтингу	+	-	-
Захист від аудіофінгерпринтингу	+	-	-
Керування cookies	+	-	+
Блокування скриптів відстеження	+	-	+
Блокування реклами	-	-	+
Аналіз вебсторінок	+	-	+
Блокування WebRTC	+	+	-
Список заблокованих доменів	+	-	+
Блокування сторонніх трекерів	-	-	+
Рандомізація цифрових відбитків браузера	+	-	-

продовження табл. 1.4

Функція	Trace	Canvas Blocker	Privacy Badger
Блокування пікселів відстеження	+	-	+
Відкритий вихідний код	+	+	+

Отже, виконавши порівняльну характеристику функціоналу інструментів для захисту від фінгерпринтингу та вебтрекінгу загалом, можемо прийти до висновку, що найбільш широкий функціонал надають розширення Trace та Privacy Badger. Проте відмінність їх полягає у тому, що Trace є потужним інструментом для спуфінгу цифрових відбитків браузера шляхом рандомізації низки атрибутів браузера, а Privacy Badger надає захист від вебвідстеження за допомогою алгоритмів штучного інтелекту, які ідентифікують та блокують трекери на основі аналізу їх поведінки. Canvasblocker надає значно менше корисного функціоналу, в порівнянні з іншими двома інструментами. Не зважаючи на те, що заявлені функції Canvasblocker виконує, проте цього недостатньо для забезпечення достатнього рівня анонімності користувача, оскільки цифрові відбитки формуються на великій кількості факторів, а Canvasblocker модифікує лише декілька з них.

Висновок до розділу 1

У першому розділі було наведено визначення вебвідстеження як методу збору інформації про користувачів в Інтернеті, описані методи, як цей процес може бути реалізований - реєстрація унікальної IP-адреси, файли cookie HTTP, створення унікального цифрового відбитку браузера, вебмаяки.

Методи відстеження можуть порушувати конфіденційність користувачів, тому в даному розділі були також описані методи запобігання вебтрекінгу та підвищення анонімності користувача, серед яких вбудований захист в браузерах Chrome, Safari та Firefox, інструменти для приховування IP-адреси – проксі-сервери, VPN та мережа TOR та програмні розширення для браузерів Privacy Badger, Canvasblocker, Trace. Був

наведений аналіз та перевірка їх функцій захисту від вебтрекінгу, наведена порівняльна характеристика.

В ході дослідження було виявлено, що найбільш інвазивним і складним для захисту з наведених методів вебвідстеження є фінгерпринтинг в браузері. Найбільш проблемним є той факт, що неможливо якимось чином видалити унікальний ідентифікатор, який сформувався, як, наприклад, файли cookie, які користувач у змозі видаляти або заблокувати.

В результаті аналізу інформації були зроблені наступні висновки: для того, щоб запобігти утворенню унікального ідентифікатора, потрібно виконати спуфінг багатьох характеристик про браузер і обладнання користувача, внаслідок чого сформується рандомізований цифровий відбиток, який допоможе браузеру користувача, по-перше, не так сильно виділятись з маси інших вебпереглядачів, а по-друге, не дозволить однозначно ідентифікувати користувача.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

2.1 Бібліотеки Python, використані під час створення програми

Програмний засіб був створений з використанням мови програмування Python, оскільки вона має великий перелік вбудованих модулів, а також додаткових, які дозволили реалізувати потрібний функціонал, і що можуть бути встановлені за допомогою пакетного менеджера pip.

У розробленій програмі були використані бібліотеки selenium, seleniumwire, requests, lxml, random, time, argparse, json, pyautogui, stem та subprocess.

Selenium — це бібліотека з відкритим кодом, яка використовується для автоматизації веббраузерів, яка забезпечує спосіб керування веббраузерами за допомогою різних мов програмування, включаючи Python. Selenium дозволяє автоматизувати такі вебвзаємодії, як натискання посилань, заповнення форм, копіювання даних тощо [37].

Seleniumwire — це розширення для Selenium, яке дозволяє перехоплювати та змінювати HTTP-запити та відповіді, зроблені браузером. Це корисно для тестування вебпрограм, які використовують AJAX, або для збирання даних із сайтів, які потребують автентифікації [38].

Requests — це бібліотека Python, яка використовується для створення HTTP-запитів. Requests абстрагує складність створення запитів за простим API, дозволяючи надсилати запити HTTP/1.1 і отримувати відповіді на них [39].

Lxml — це бібліотека Python для роботи з документами XML і HTML, що забезпечує швидкий і простий у використанні інтерфейс для аналізу та обробки документів XML і HTML, а також підтримує селектори XPath і CSS для вибору елементів [40].

Модуль Random — це бібліотека Python, яка використовується для генерації випадкових чисел. Він надає функції для генерування випадкових цілих чисел, чисел з плаваючою точкою та послідовностей [41].

Модуль Time — це бібліотека Python, яка використовується для вимірювання часу та обробки дат і часу. Він надає функції для отримання поточного часу, переходу в режим сну на задану кількість секунд і перетворення між різними форматами часу [42].

Argparse — це бібліотека Python для аналізу аргументів командного рядка, що полегшує створення зручних інтерфейсів командного рядка та забезпечує вбудовану підтримку багатьох поширених типів аргументів, таких як рядки, цілі числа та логічні значення [43].

Модуль json — це бібліотека Python, яка використовується для роботи з даними JSON. Він надає функції для кодування об'єктів Python як рядків JSON і декодування рядків JSON в об'єкти Python [44].

Pyautogui — це бібліотека Python, яка використовується для автоматизації завдань GUI. Він надає функції для імітації дій миші та клавіатури, створення знімків екрана та керування вікнами [45].

Stem — це бібліотека Python, яка використовується для взаємодії з мережею TOR, яка забезпечує спосіб програмного керування TOR і використання його як проксі-сервера для надсилання анонімних запитів [46].

Модуль subprocess — це бібліотека Python, яка використовується для виконання зовнішніх команд і спілкування з ними, надає функції для створення нових процесів, надсилаючи їм вхідні дані та фіксуючи їх вихідні дані [47].

2.2 Функціонал програмного засобу

Програмний засіб розроблений для браузера Google Chrome, оскільки, як було виявлено в попередньому розділі, він хоч і є найпоширенішим серед користувачів, але в порівнянні з іншими сучасними вебпереглядачами, має невисокий рівень захисту від вебтрекінгу та фінгепринтингу в браузері зокрема. Тож програма представляє собою

консольний додаток, що виконує функції захисту від фінгерпринтингу шляхом спуфінгу параметрів браузеру користувача і має низку параметрів для налаштування.

Сценарій Python пропонує наступні функції:

- відключення JS;
- зміна заголовку User-Agent на випадковий;
- приховування мов інтерфейсу;
- використання режиму інкогніто;
- відключення cookies;
- налаштування розміру вікна;
- використання проксі-серверів та Tor для приховування IP;
- блокування WebRTC.

На рис. 2.1 представлена діаграма, в якій показані варіанти використання програмного засобу користувачем.

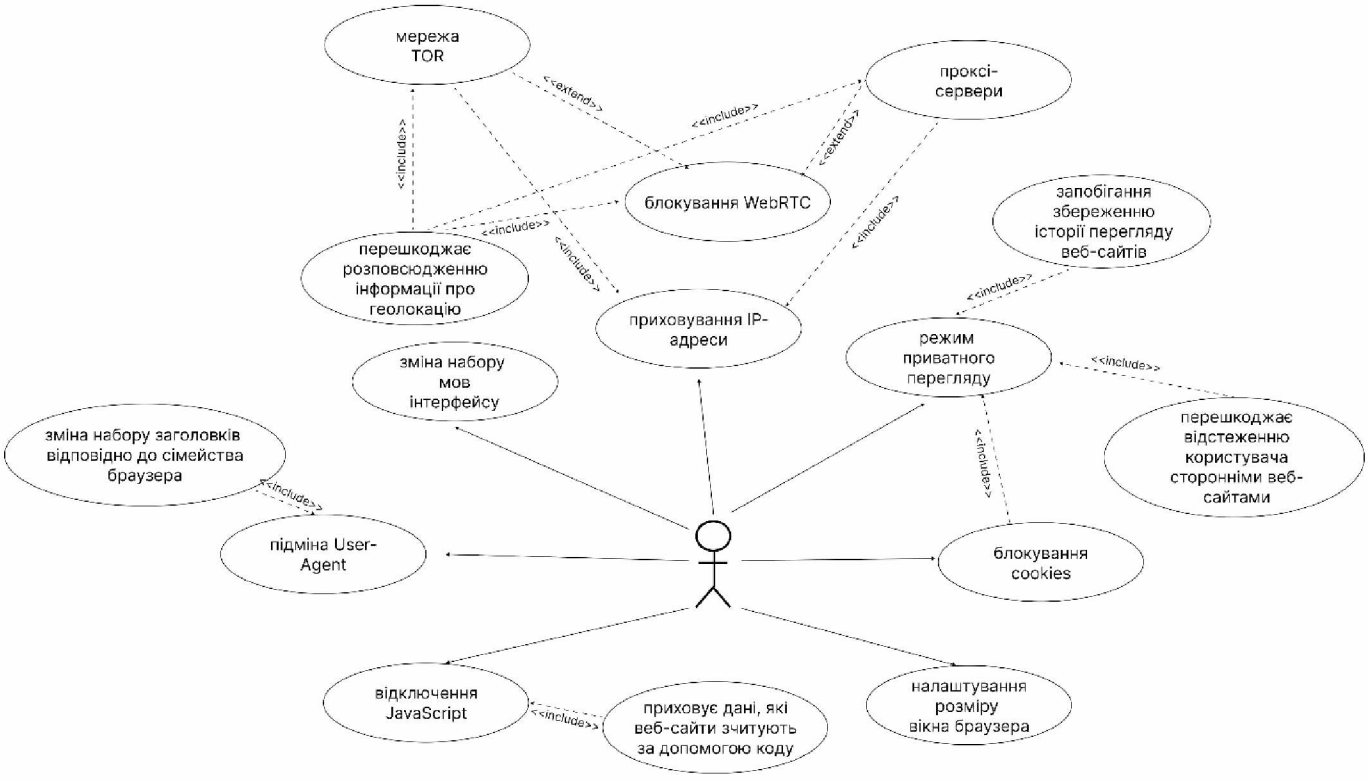

Рисунок 2.1 – Діаграма прецедентів

Скрипт використовує параметри командного рядка, які задаються користувачем, відповідно до налаштувань конфігурації браузера Chrome, які він бажає застосувати. Параметри включають наступне:

- -js або --javascript – відключення JavaScript
- -ua або --useragent – зміна User-Agent на випадковий
- -wW або --winWidth – налаштування ширини вікна
- -wH або --winHeight – налаштування висоти вікна
- -l або --languages – приховування мов інтерфейсу
- -c або --cookies – відключення cookies
- -i або --incognito – включення режиму інкогніто
- -p або --proxy – використання проксі для приховування IP
- -t або --tor – використання Tor для приховування IP.

На рис. 2.2 наведена довідка, що описує використання програми.

```
┌──(kali㉿kali)-[~/Documents]
└─$ python3 antitracking.py -h
usage: Antifingerprinting module [-h] [-js] [-ua] [-wW WINWIDTH] [-wH WINHEIGHT] [-l LANGUAGES] [-c] [-i] [-p]
                                 [-t]

**Program Functionality**

options:
  -h, --help            show this help message and exit
  -js, --javascript     disable JS
  -ua, --useragent      switch to a random User-Agent
  -wW WINWIDTH, --winWidth WINWIDTH
                        customize window width
  -wH WINHEIGHT, --winHeight WINHEIGHT
                        customize window height
  -l LANGUAGES, --languages LANGUAGES
                        hide interface languages; argument must be passed in format "uk-UA, de-DE, es-ES"
  -c, --cookies         disable cookies
  -i, --incognito       enable incognito mode
  -p, --proxy           use Proxy to hide your IP
  -t, --tor             use Tor to hide your IP; you must start Tor service before launch this script

This program was created by (c)Kyselova Alina
```

Рисунок 2.2 – Help: функціонал програми

Варто зазначити, що разом із заголовком User-Agent змінюються і інші заголовки, які відповідають обраному агенту користувача, оскільки порядок і кількість полів заголовків HTTP є унікальними для кожного сімейства браузерів.

Якщо застосований аргумент --tor, програма надсилає сигнал мережі TOR про зміну IP-адреси через регулярні проміжки часу, тобто реалізується ротація IP-адреси.

Для коректної роботи програми необхідно перед її запуском ввімкнути службу TOR, якщо планується використовувати мережу TOR для приховання IP-адреси.

2.3 Алгоритм роботи програми

На рис. 2.3 наведена блок-схема, що графічно демонструє алгоритм роботи програми.

Рисунок 2.3 – Алгоритм роботи програми

Основні кроки алгоритму програми можна описати наступним чином:

1. Початок програми. Імпорт необхідних бібліотек.

2. Обробка параметрів командного рядка, що використовуються для налаштування функціоналу програми.

3. Налаштування веббраузера. Програма налаштовує веббраузер Chrome з використанням опцій та налаштувань, які відповідають переданим параметрам командного рядка.

4. Запуск веббраузера. Програма запускає веббраузер Chrome з відповідними налаштуваннями.

5. Відвідування вебсторінок. Користувач може відвідувати різні вебсторінки та взаємодіяти з ними.

6. Завершення роботи програми. Після завершення взаємодії з вебсторінками програма завершує свою роботу.

Висновок до розділу 2

В другому розділі було виконане проєктування програмного засобу, призначеного для забезпечення захисту користувачів мережі Інтернет від фінгерпринтингу в браузері та приховування реальних їх даних від вебсайтів, трекерів та інших сервісів, які можуть збирати дані про користувача, натомість надаючи випадкові дані(IP-адреси, UA, розмір вікна) суб'єктам, які роблять спроби утворення цифрових відбитків.

Програма має перелік параметрів командного рядка, що були описані в даному розділі і використовуються для налаштування функціоналу програми. Розроблений інструмент володітиме інструментарієм для відключення JavaScript, використання випадкового UA, встановлення розміру вікна браузера, відключення файлів cookies, використання безкоштовних анонімних проксі-серверів або мережі TOR для приховання IP-адреси. Програма також реалізує функції парсингу та перевірки проксі-серверів на працездатність.

Загальна логіка роботи програми полягає в налаштуванні браузера Chrome з використанням опцій та налаштувань, які відповідають переданим параметрам командного рядка, та запуску браузера з відповідними налаштуваннями.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Опис функцій програмного коду

Функція parse_arguments() аналізує аргументи командного рядка, створює парсер за допомогою бібліотеки argparse, використовуючи метод ArgumentParser(). Метод add_argument() додає параметри командного рядка до парсера. Кожен параметр має свій флаг (-js, -ua тощо) і текстовий опис функції параметра, що виводиться у довідці програми. Метод parse_args() запускає парсер і розміщує отримані дані в об’єкт argparse.Namespace. Отже, як результат виконання функція parse_arguments() повертає об'єкт, який містить атрибути, що відповідають значенням параметрів командного рядка. Значення параметрів встановлюються користувачем при запуску програми через термінал.Лістинг коду функції на рис. 3.1.

```
def parse_arguments():
    parser = argparse.ArgumentParser(
        prog='Antifingerprinting module',
        description='**Program Functionality**',
        epilog='This program was created by (c)Kyselova Alina')

    parser.add_argument(
        '-js', '--javascript', action='store_true', help='disable JS')
    parser.add_argument(
        '-ua', '--useragent', action='store_true', help='switch to a random User-Agent')
    parser.add_argument(
        '-wW', '--winWidth', action='store', help='customize window width')
    parser.add_argument(
        '-wH', '--winHeight', action='store', help='customize window height')
    parser.add_argument(
        '-l', '--languages', action='store',
        help='hide interface languages; argument must be passed in format "uk-UA, de-DE, es-ES"')
    parser.add_argument(
        '-c', '--cookies', action='store_true', help='disable cookies')
    parser.add_argument(
        '-i', '--incognito', action='store_true', help='enable incognito mode')
    parser.add_argument(
        '-p', '--proxy', action='store_true', help='use Proxy to hide your IP')
    parser.add_argument(
        '-t', '--tor', action='store_true',
        help='use Tor to hide your IP; you must start Tor service before launch this script')

    return parser.parse_args()
```

Рисунок 3.1 – Лістинг коду функції parse_arguments()

Функція get_proxies(r) виконує парсинг списку безкоштовних проксі-серверів з сайту free-proxy-list.net. Функція приймає аргумент r, який є кількістю проксі-серверів для повернення. Як видно на рисунку, код реалізований так, що функція зчитує значення HTML-елементів і додає потрібні (тобто IP-адреси проксі-серверів) до списку proxies і повертає цей список. Лістинг коду функції на рис. 3.2.

```
def get_proxies(r):
    url = 'https://free-proxy-list.net/'
    response = requests.get(url)
    parser = fromstring(response.text)
    proxies = set()
    for i in parser.xpath('//tbody/tr')[:r]:
        if i.xpath('.//td[7][contains(text(),"yes")]'):
            proxy = ":".join([i.xpath('.//td[1]/text()')[0], i.xpath('.//td[2]/text()')[0]])
            proxies.add(proxy)

    return proxies
```

Рисунок 3.2 – Лістинг коду функції get_proxies

Функція check_proxies(proxies) приймає аргумент proxies, який містить список проксі-серверів, які були отримані за допомогою функції get_proxies() і повертає список дійсних проксі-серверів. Доступність проксі-сервери перевіряються шляхом надсилання запиту get на адресу google.com: якщо код статусу у відповіді «200», тоді робимо висновок, що даний сервер працює і додаємо до списку valid_proxies. В іншому випадку – відбувається перехід до наступного у списку проксі-сервера. Коли весь список proxies перевірений на доступність, функція завершує своє виконання і повертає список valid_proxies. Лістинг коду функції на рис. 3.3.

```
def check_proxies(proxies):
    valid_proxies = []
    for proxy in proxies:
        try:S
            response = requests.get('https://www.google.com/', proxies={'http': proxy, 'https': proxy}, timeout=10)
            if response.status_code == 200:
                valid_proxies.append(proxy)
            else:
                pass
        except:
            pass
    return valid_proxies
```

Рисунок 3.3 – Лістинг коду функції check_proxies

Функція read_data_json() виконує читання файлу JSON, що містить інформацію про заголовки HTTP, значення яких встановлюються при зміні агента користувача. Лістинг коду функції наведений на рис. 3.4. В файлі headers.json (рис. 3.5) міститься об'єкт JSON з наборами заголовків для різних браузерів. У даній функції вміст зазначеного файлу зчитується у змінну data за допомогою функції load бібліотеки json і read_data_json в результаті виконання повертає змінну data.

```
def read_data_json():
    with open('headers.json') as f:
        data = json.load(f)

    return data
```

Рисунок 3.4 – Лістинг коду функції read_data_json


```
        "upgrade-insecure-requests": "1",
        "user-agent": "Mozilla/5.0 (Windows NT 10.0; Windows; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
103.0.5060.114 Safari/537.36",
        "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
*;q=0.8,application/signed-exchange;v=b3;q=0.9",
        "sec-ch-ua": "\".Not/A)Brand\";v=\"99\", \"Google Chrome\";v=\"103\", \"Chromium\";v=\"103\"",
        "sec-ch-ua-mobile": "?0",
        "sec-ch-ua-platform": "\"Windows\"",
        "sec-fetch-site": "none",
        "sec-fetch-mod": "",
        "sec-fetch-user": "?1",
        "accept-encoding": "gzip, deflate, br",
        "accept-language": "bg-BG,bg;q=0.9,en-US;q=0.8,en;q=0.7"
    },
```


Рисунок 3.5 – Набір HTTP заголовків

Функція interceptor(request) - перехоплювач запитів, він є вкладеною фуенкцією всередині функції set_preferences і викликається об'єктом seleniumwire.webdriver з функції set_preferences для зміни заголовків вхідних запитів у випадку, якщо був використаний прапор -ua (для зміни User-Agent). Функція видаляє заголовок User-Agent та замінює цей та інші заголовки на випадковий набір заголовків зі словника d. Лістинг коду функції на рис. 3.6.

```
    def interceptor(request):
        del request.headers["user-agent"]

        for header in d.keys():
            request.headers[f"{header}"] = d[f"{header}"]
```

Рисунок 3.6 – Лістинг коду функції interceptor

Функція set_preferences(options, preferences, args) змінює параметри веббраузера. Вона приймає три аргументи: “options”, який є об’єктом selenium.webdriver.chrome.options.Options; “preferences”, який є словником, що містить налаштування браузера; і “args”, який є об’єктом argparse.Namespace, що містить аргументи командного рядка. Функція змінює параметри та об’єкти налаштувань на основі аргументів, переданих програмі. Логіка даної функції полягає у тому, що вона перевіряє, чи містить змінна args певні параметри командного рядка, наприклад -proxy, -languages, -cookies, -winWidth і -winHeight тощо. Якщо так, він встановлює відповідні параметри за допомогою словника параметрів, який передається до змінної options. Якщо змінна args містить параметр -proxy, функція отримує список проксі-серверів із функції get_proxies(), перевіряє їх за допомогою функції check_proxies() і вибирає випадковий дійсний проксі-сервер для використання у браузері. Загальна логіка опрацювання переданих користувачем аргументів реалізована у функції set_preferences() і саме в ній встановлюються всі конфігурації браузера.

Алгоритм роботи функції set_preferences(options, preferences, args):

Якщо встановлений параметр --tor, то:

- відкривається з'єднання з контролером TOR, яке прослуховує на порту 9051;
- після успішної аутентифікації за допомогою пароля “abc123” (пароль може відрізнятись в залежності від локальним налаштувань системи), відправляється сигнал NEWNYM до контролера, щоб сервіс TOR змінив IP-адресу для вихідного з'єднання;
- створюється словник options_s, який містить налаштування для проксі-сервера, який маршрутизує мережевий трафік через локальний TOR-сервер за

допомогою протоколу SOCKS5. Він буде використовуватись для налаштування сесії в браузері;

2. Якщо встановлений параметр --proxy, то

- створюється список proxy;
- запускається цикл while, в якому викликатиметься функція для парсингу IP-адрес безкоштовних проксі серверів (get_proxies(i)), які додаються до списку proxies. За один крок циклу робиться запит на i=15 адрес, проте якщо список пустий, то робиться запит на наступні 15 адрес;
- коли адреси були збережені в список proxies, робота циклу завершується і проводиться перевірка на доступність цих проксі (викликається функція check_proxies(proxies)). Результат роботи check_proxies зберігається у змінну valid_proxies;
- випадковий проксі сервер обирається за допомогою функції choice бібліотеки random зі списку valid_proxies і це значення зберігається у змінну proxy;
- до конфігурацій браузера додаємо проксі-сервер: *options.add_argument(f'--proxy-server={proxy}');*
- оновлюється словник preferences для того, щоб додати функцію блокування WebRTC:

preferences.update({"webrtc.ip_handling_policy":"disable_non_proxied_udp"})

preferences.update({"webrtc.multiple_routes_enabled": False})

preferences.update({"webrtc.nonproxied_udp_enabled": False})

3. Якщо встановлений параметр --languages, то оновлюється словник preferences, щоб встановити бажані мови інтерфейсу в браузері: *preferences.update({"intl.accept_languages": f"{args.languages}"})*

4. Якщо встановлений парамет --cookies, то оновлюється словник preferences, щоб заблокувати використання файлів cookie:

preferences.update({“profile.default_content_settings.cookies”: 2})

preferences.update({“profile.block_third_party_cookies”: True}

preferences.update({“profile.default_content_setting_values.cookies”: 2})

5. Якщо встановлені параметри --winWidth або –winHeigth, то виконується:

- зчитування розміру екрану за допомогою методу pyautogui.size();
- встановлюється значення булевої змінної flag у значення True;
- за допомогою конструкції try…catch виконуємо конвертацію переданих значень аргументів командного рядку у тип int. Якщо ковертація не виконується, то відбувається обробка виключення і flag присвоюється значення False;
- якщо flag = True, перевіряються декілька умов і встановлюється нова конфігурація браузера, що визначає розмір вікна:

options.add_argument(f'--window-size={args.winWidth},{args.winHeight}')

6. Якщо встановлений параметр --javascript, то оновлюємо словник preferences, щоб вимкнути використання Javascript на вебсторінках:

preferences.update({"profile.managed_default_content_settings.javascript": 2})

7. Якщо встановлений параметр --incognito, то додаємо аргумент "*--incognito*" до об'єкта options, щоб ввімкнути анонімний режим перегляду:

options.add_argument("--incognito")

8. До об'єкту options додається в якості аргумента словник preferences, в якому містяться нові конфігурації браузера:

options.add_experimental_option("prefs", preferences)

9. Якщо встановлений параметр args.useragent, то:

- викликається функція read_data_json(), що зчитує набори заголовків з різними User-Agent з файлу JSON і зберігається у змінну data (data – список словників);
- вибирається випадковий набір заголовків із збережених у змінній data і обраний набір зберігається у змінній d (d – словник, що містить пари ключ-значення);
- змінній user_agent присвоюється значення ключа 'user-agent' зі словника d;
- встановлюється перехоплювач запитів interceptor, який видаляє заголовок User-Agent, встановлює новий, що міститься у змінній user_agent і за допомогою цикла for перезаписує значення або додає всі інші HTTP заголовки, що містяться в словнику d.

10. Створюється екземпляр об'єкту вебдрайвера Chrome з певними опціями (збережені в змінній options) і параметрами Selenium Wire (збережені в змінній options_s):

driver=webdriver.Chrome(options=options, seleniumwire_options=options_s)

В результаті виконання функції set_preferences повертається раніше встановлений екземпляр об'єкту вебдрайвера Chrome driver з новими конфігураціями.

Графічно логіку роботи функції set_preferences() наведено на блок-схемі, зображеній на рис. 3.7-3.8.

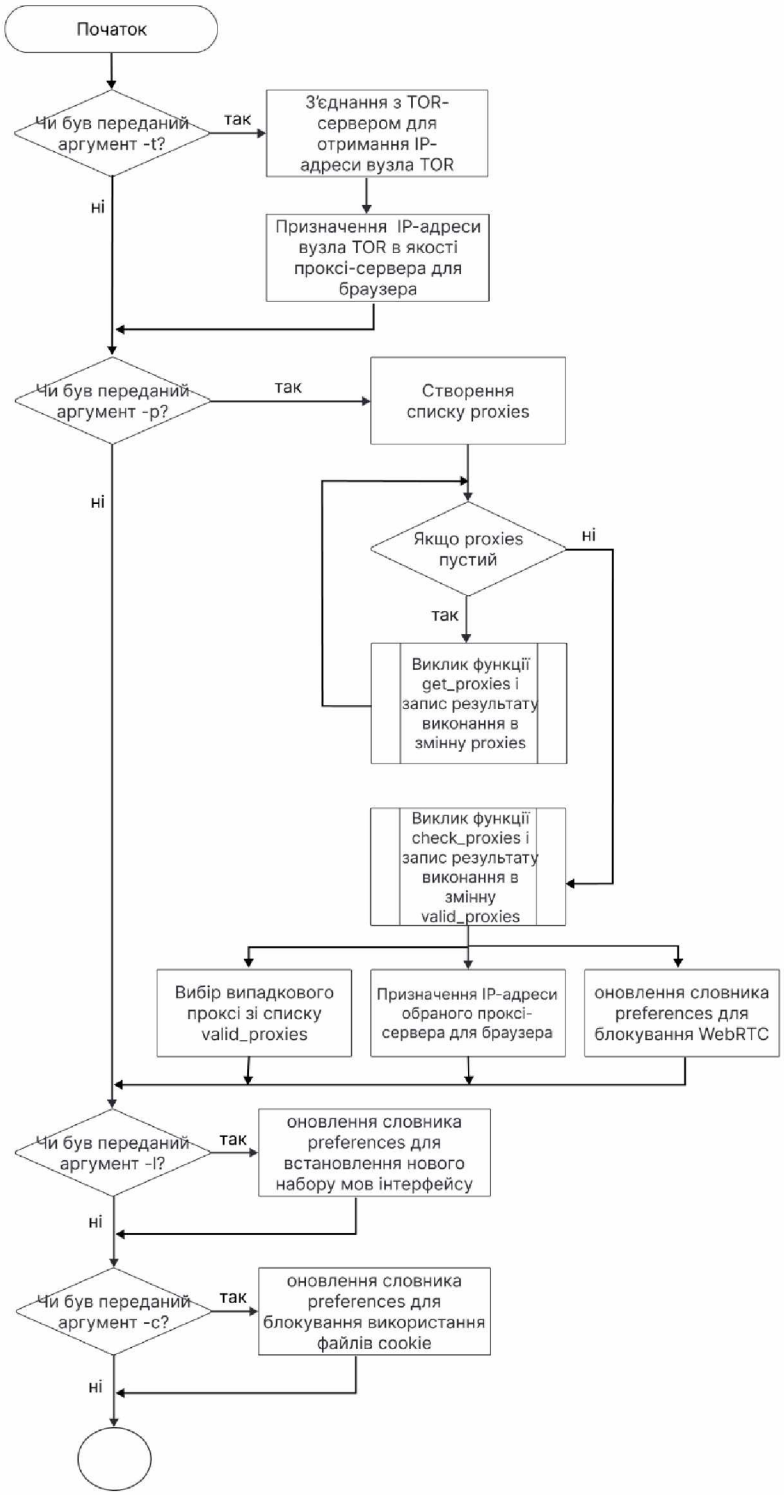

Рисунок 3.7 – Блок-схема функції set_preferences

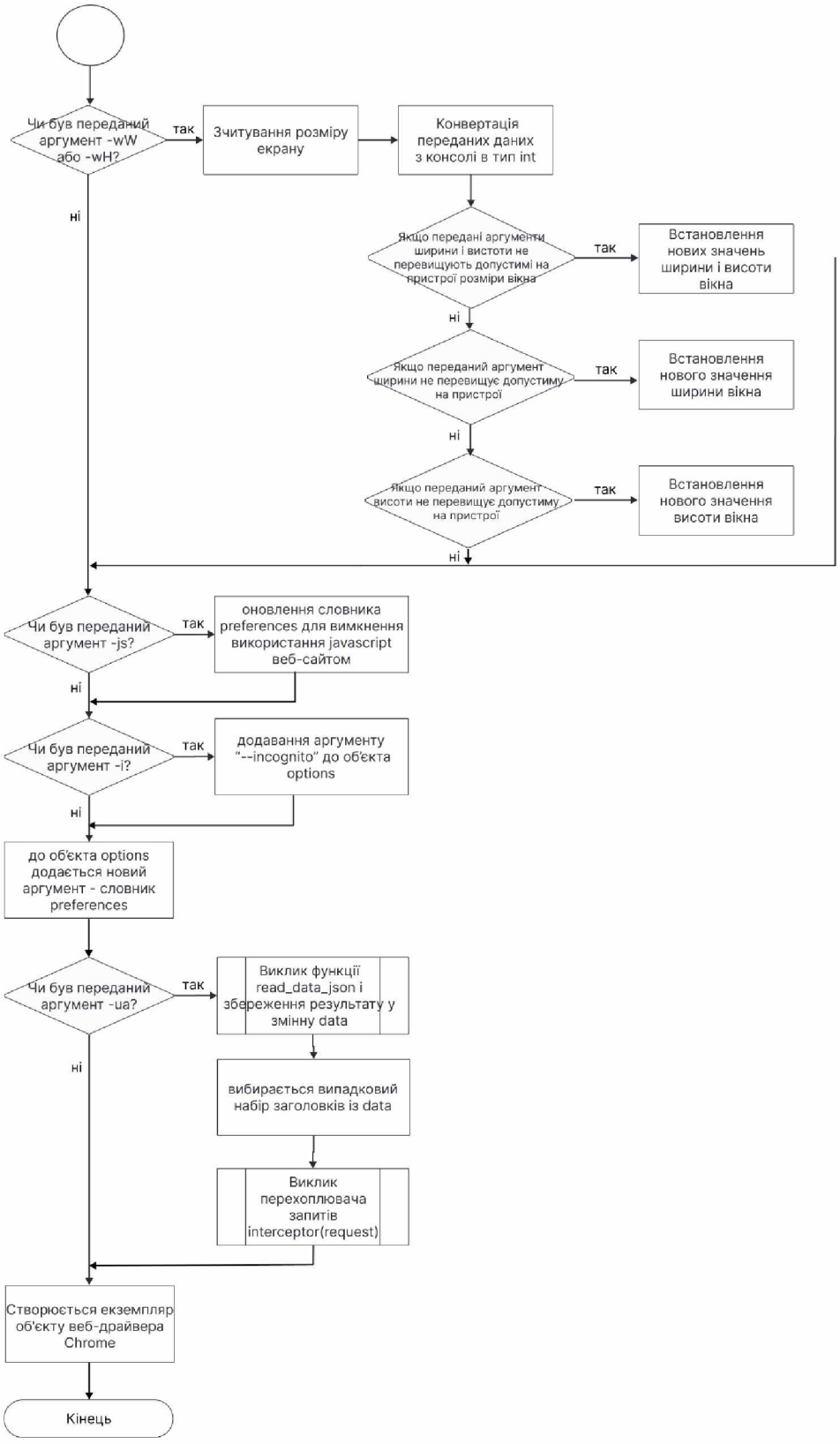

Рисунок 3.8 – Блок-схема функції set_preferences(продовження)

Функція main() є основною частиною програми, з якої викликаються функції parse_arguments та set_parameters. В даній функції створюється об’єкт seleniumwire.webdriver зі модифікованими налаштуваннями браузера, за допомогою

змінної options ініціалізується драйвер Chrome та завантажує вебсайт за вказаною URL-адресою - http://httpbin.org/anything, яка показує інформацію про деякі конфігурації браузера (IP-адреса, заговоловки). Користуючись запущеним вікном браузера Chrome, користувач підвищить рівень власної анонімності в мережі, оскільки у створеного екземпляра об'єкту вебдрайвера Chrome будуть ті конфігурації, які були налаштовані безпосередньо користувачем за допомогою розробленого інструменту, а не ті, що надаються вебпереглядачем за замовчуванням. На рис. 3.9 наведена блок-схема функції main, яка графічно описує логіку роботи програми.

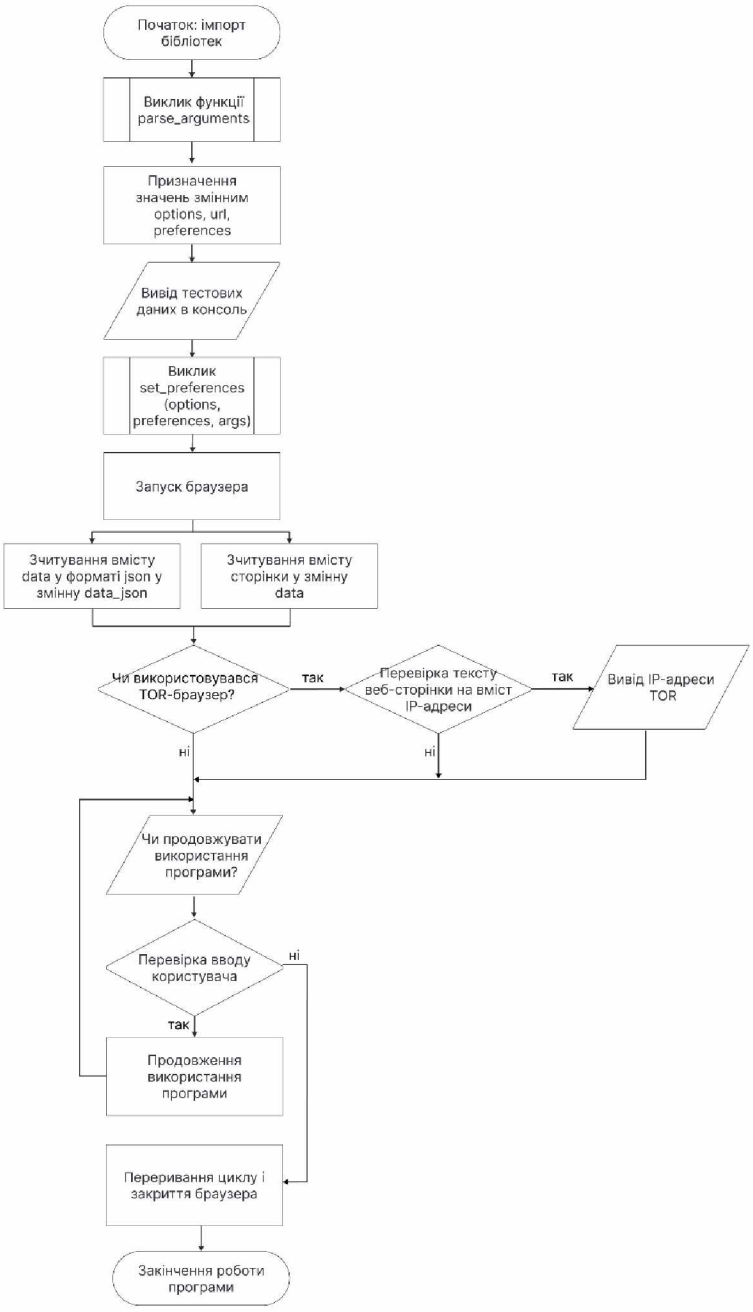

Рисунок 3.9 – Блок-схема функції main

Змінні на блок-схемі:

- url – адреса сторінки, яка буде відкриватись при запуску браузера. URL-адреса - httpbin.org/anything показує інформацію про деякі конфігурації браузера (IP-адреса, заговоловки тощо);
- options – клас для керування параметрами ChromeDriver;
- preferences – словник, що містить конфігурації для браузера;
- args – аргументи командного рядка, що були передані користувачем при запуску програми;
- data – тескстові дані, отримані зі сторінки *httpbin.org/anything*;
- data_json – дані зі змінної data, конвертовані в формат json;

На рис. 3.10 графічно показана схема зв'язків між функціями і файлами.

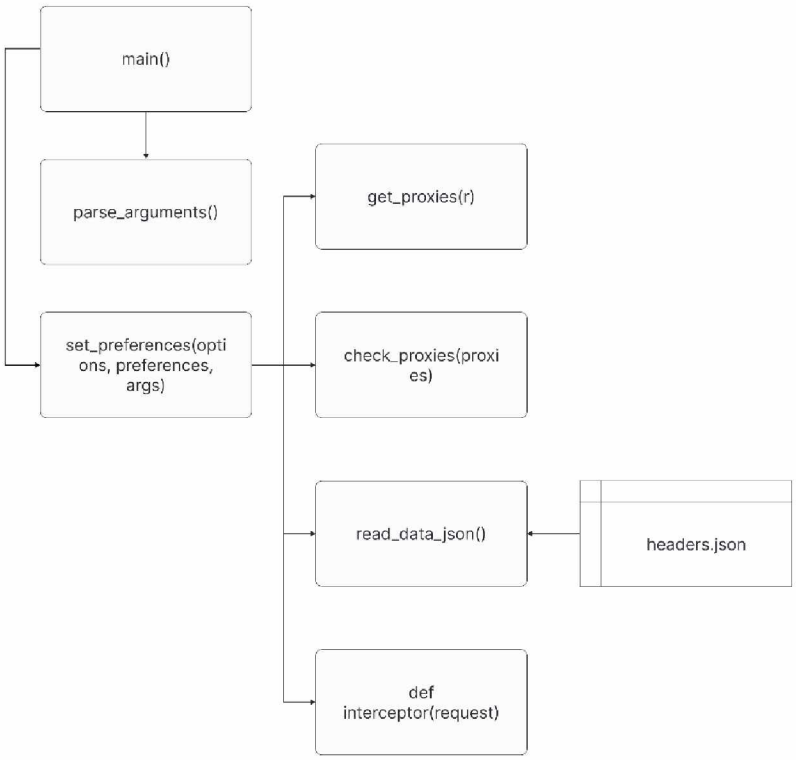

Рисунок 3.10 – Схема зв'язків функцій і файлів

Загалом, розроблений програмний засіб надає гнучкий спосіб налаштувати перегляд вебсторінок і зменшити ризик однозначної ідентифікації користувача на основі цифрових відбитків браузера на вебсайтах.

3.2 Запуск програми в термінальному середовищі

Як було зазначено у попередньому розділі, розроблена програма є консольним додатком, тому її запуск відбувається з командного рядка. Для того, щоб ознайомитись з функціоналом програмного засобу, користувач має запустити скрипт із аргументом --help або -h (рис. 3.11)

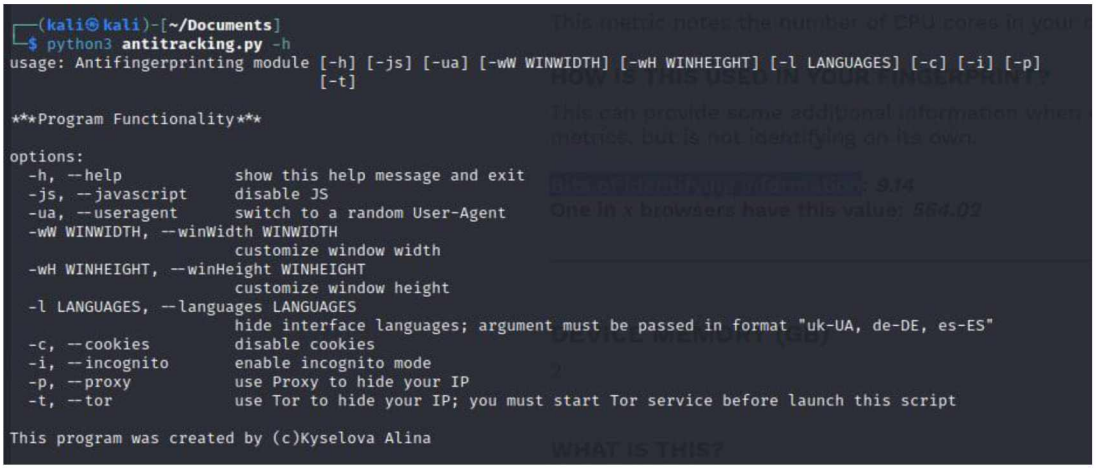

Рисунок 3.11 – Довідка програми

Ознайомившись із функціоналом, користувач може запустити програму на виконання, щоб налаштувати потрібні йому конфігурації Google Chrome, передавши відповідні аргументи командного рядка, які були описані у другому розділі (рис. 3.12).

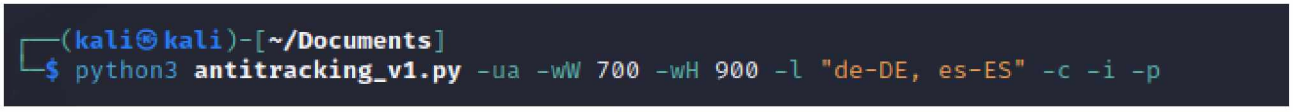

Рисунок 3.12 – Запуск програми з довільними аргументами командного рядка

Після запуску у виводі програма надасть інформацію про встановлені користувачем нові конфігурації вебпереглядача (рис. 3.13)

Рисунок 3.13 – Вивід програми: налаштовані користувачем конфігурації

Через 10 секунд (цей часовий використовується в програмі за замовчуванням) з'явиться запит до користувача, чи продовжувати роботу у браузері з налаштованими конфігураціями для захисту, чи завершити виконання програми і роботу у вебпереглядачі (рис. 3.14).

Рисунок 3.14 – Запит на продовження або припинення роботи

У випадку, якщо користувач або не вводить нічого, або вводить "Y" або"y", виконання програми продовжується і через таймаут, що дорівнює 200 секунд знову з'явиться запит на продовження чи припинення роботи. Якщо він вводить будь-який інший символ або рядок, то виконання програми припиняється і відбувається вихід з браузера (рис. 3.15).

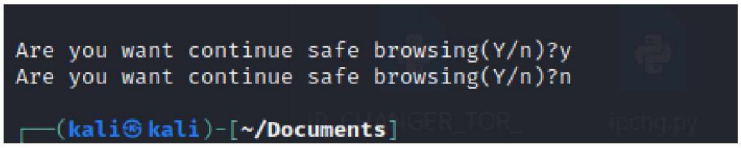

Рисунок 3.15 – Продовження роботи програми/вихід із програми

3.3 Тестування програмного засобу

Тестування вебпереглядача з використанням ресурсу DeviceInfo.me

Сервіс DeviceInfo.me [48] є онлайн-інструментом, який надає користувачам інформацію про їхні пристрої, зокрема браузер, операційну систему, IP-адресу та інші технічні деталі. Основні функції цього сервісу включають відображення технічної інформації про пристрій – DeviceInfo.me надає інформацію про тип та версію використовуваного браузера, операційну систему (включаючи назву та версію), розширення браузера, підтримувані технології (наприклад, Flash, Java, Cookies тощо); визначення IP-адреси – DeviceInfo.me може показати IP-адресу, з якої користувач взаємодіє з сервісом; тестування пристрою – DeviceInfo.me дозволяє виконати деякі тести, які допомагають визначити характеристики пристрою.

Результати тестування веббраузера без використання розробленого програмного засобу наведені на рис. 3.16 – 3.21.

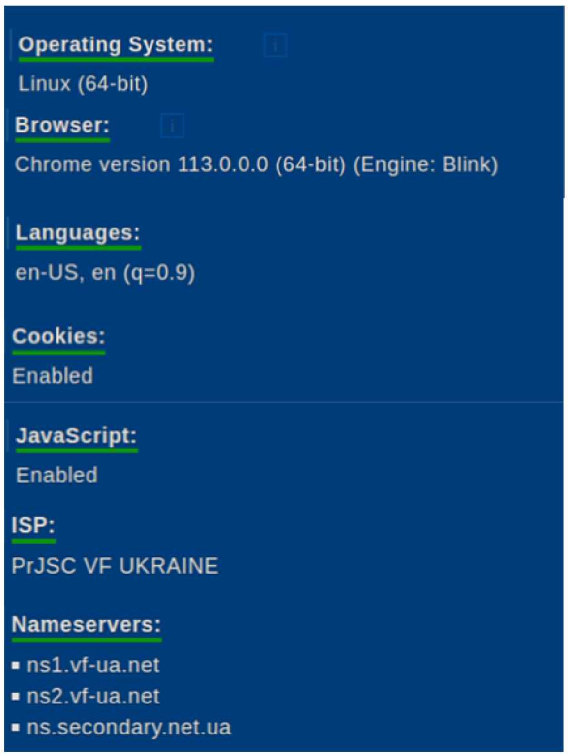

Рисунок 3.16 – Операційна система, версія браузера, мова інтерфейсу, стан файлів cookie, стан javascript, інформація про ISP

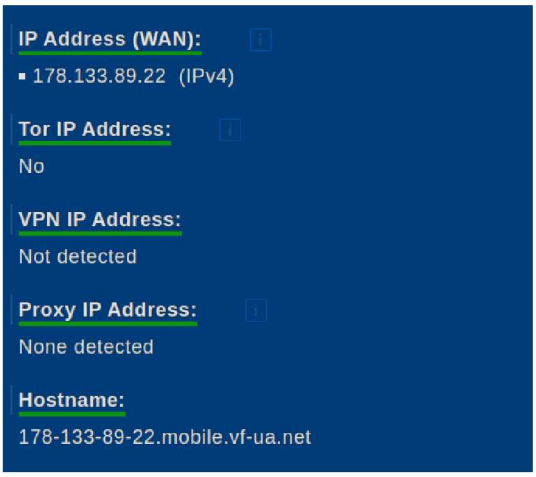

Рисунок 3.17 – Інформація про IP-адресу

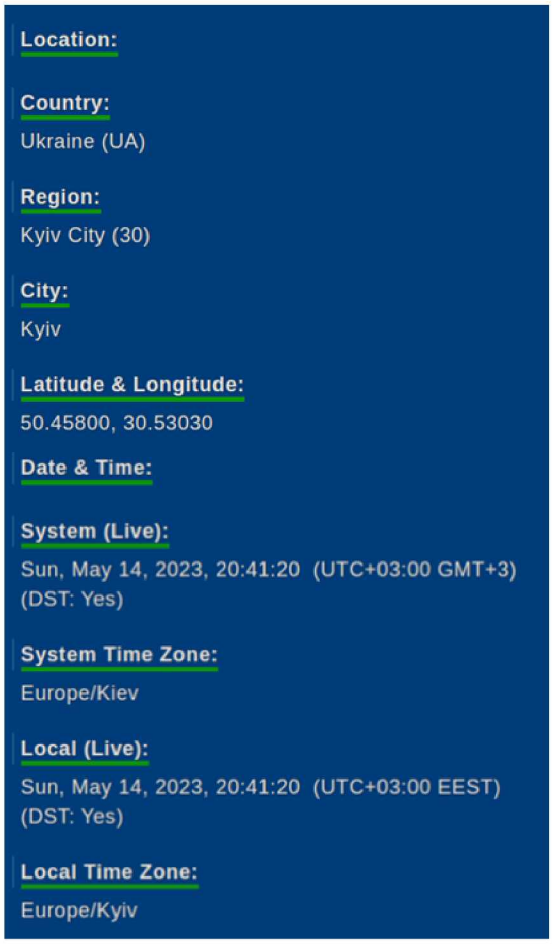

Рисунок 3.18 – Геолокація, дата та час

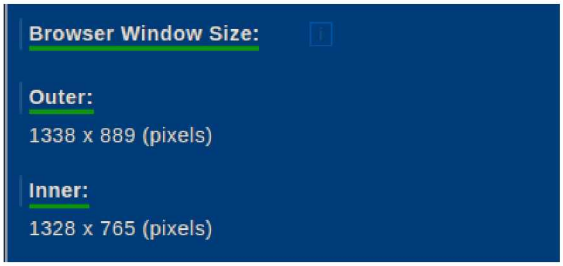

Рисунок 3.19 – Розмір вікна браузера

Рисунок 3.20 – Заголовок User-Agent

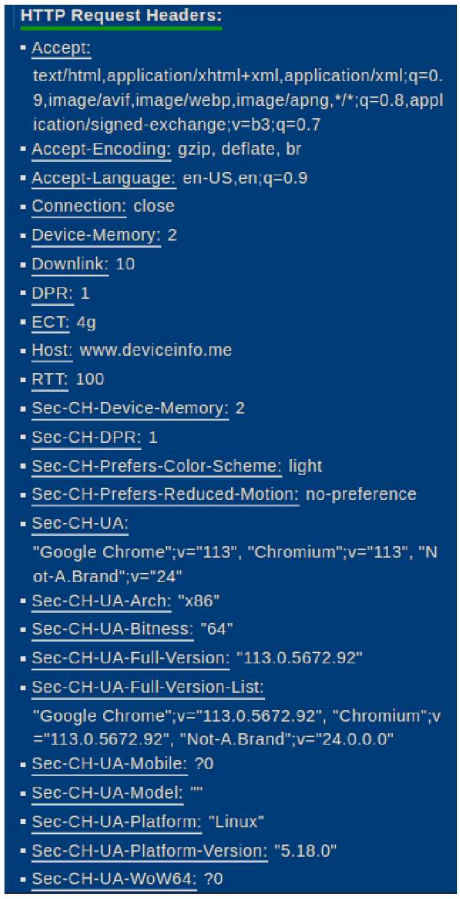

Рисунок 3.21 – Інші заголовки HTTP

Для порівняння результатів, запустимо розроблений ПЗ та переглянемо, як змінюються конфігурації вебпереглядача. В загальному випадку, щоб досягти кращого ефекту від застосування даного застосунку для запобігання фінгерпринтингу і підвищення рівня анонімності користувача, рекомендується застосовувати якомога більше параметрів. Це пояснюється тим, що цифровий відбиток формується з великої кількості різних конфігурацій браузера і чим більше з них будуть змінені, тим більша ймовірність того, що веббраузер користувача буде більш захищений.

Проте, існують такі параметри, зміна яких навпаки спровокує те, що веббраузер користувача буде більш вирізнятися із низки інших з тої причини, що значення певного параметру буде нехарактерним для більшості. Прикладом такого параметру є

відключення Javascript, оскільки у більшості вебпереглядачів він за замовченням увімкнений, оскільки це сприяє коректній роботі вебсайтів.

Отже, спочатку спробуємо запустити скрипт на виконання із аргументами командного рядка -t, -c, -i, -js, -l, ua, wW, wH (рис. 3.22), що ввімкнуть зміну IP-адреси за допомогою TOR-мережі, вимкнуть використання файлів cookie, використання JS, змінить мови інтерфейсу, агент користувача та розмір вікна браузера. Результати тестування наведені на рис. 3.23-3.30.

```
┌──(kali㉿kali)-[~/Documents]
└─$ python3 antitracking_v1.py -t -c -i -js -ua -wW 700 -wH 900 -l "de-DE, es-ES"
```

Рисунок 3.22 – Запуск програми з аргументами -t, -c, -i, -js, -l, ua, wW, wH

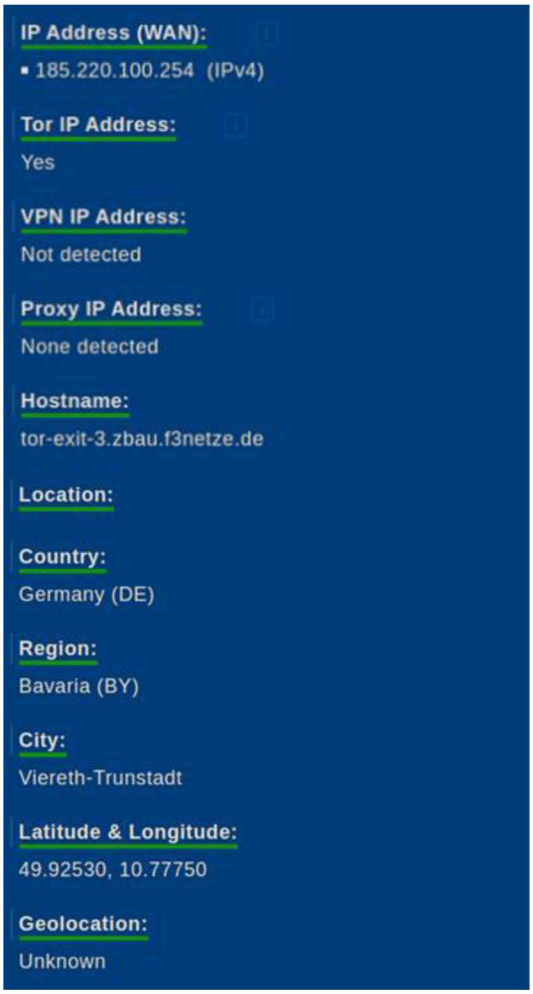

Рисунок 3.23 – Інформація про IP-адресу (використання мережі TOR), геолокація

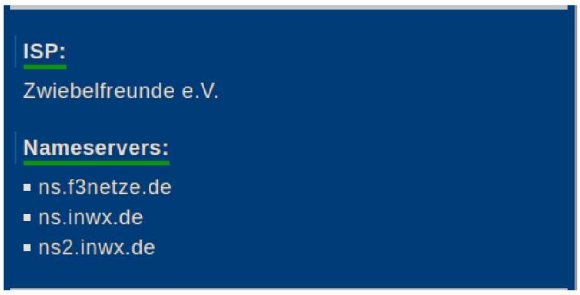

Рисунок 3.24 – Інформація про ISP

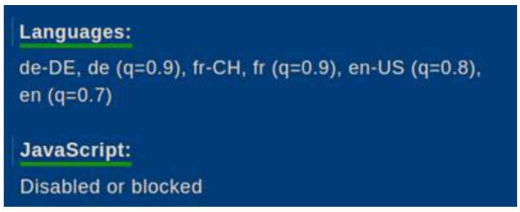

Рисунок 3.25 – Мова інтерфейсу, стан javascript

Рисунок 3.26 – Розмір вікна браузера

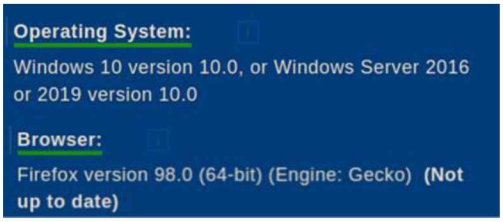

Рисунок 3.27 – Операційна система, версія браузера

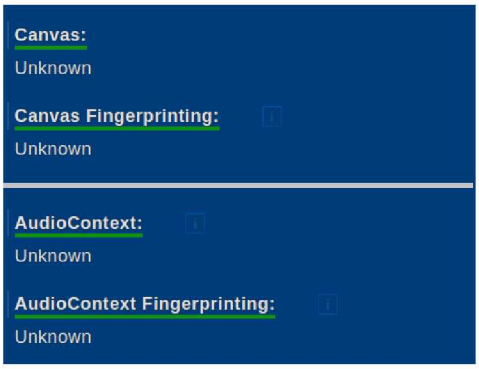

Рисунок 3.28 – Інформація про Canvas та AudioContext фінгерпринтинг

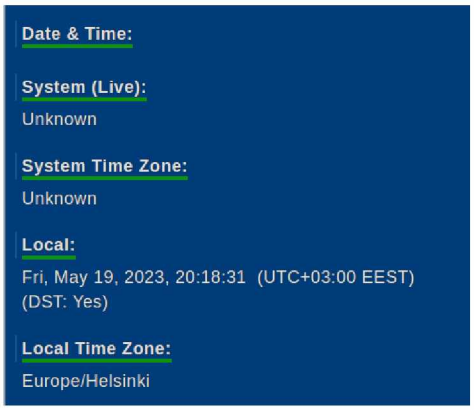

Рисунок 3.29 – Дата і час

Рисунок 3.30 – Заголовки HTTP

Як ми бачимо, була змінена IP-адреса пристрою на адресу, яка належить мережі TOR (сервіс додає помітку, що використовується адреса вузла TOR.), відповідно до нової IP-адреси змінилась геолокація, наведена інформація про нового постачальника інтернет-послуг, локальний час визначається відповідно до розташування IP-адреси, а системний час не зчитується, змінений перелік мов інтерфейсу, відключений JS. Блокується інформація про такі параметри як інформація про розмір вікна браузера, стан батареї, Canvas, WebGL, AudioContext, шрифти, стан файлів cookie, RAM, розширення браузера тощо.

Якщо застосувати параметр -js, то багато інформації стає заблоковано для зчитування. Але якщо не використовувати цей параметр, то буде виводитись інформація, що наведена на рис. 3.31 – 3.36. При ввімкненому Javascript також стає

доступною інформація про розмір вікна браузера, Canvas, WebGL, AudioContext, шрифти, стан файлів cookie, RAM, розширення браузера. А також у заголовках наводиться інформація і про агент користувача, який використовується фактично, і про той, що був встановлений за допомогою розробленого програмного засобу.

Рисунок 3.31 – Заголовок User-Agent

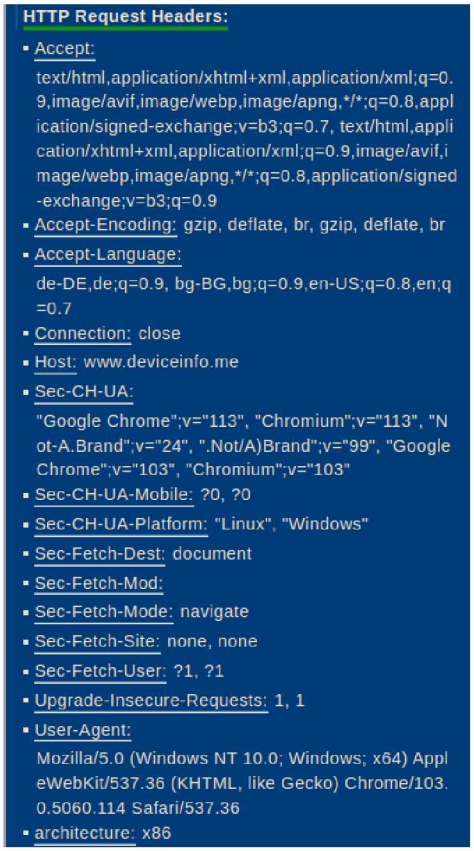

Рисунок 3.32 – Інші заголовки HTTP

Рисунок 3.33 – Розмір вікна браузера

Рисунок 3.34 – Стан javascript, стан файлів cookie

Рисунок 3.35 – Інформація про Canvas та AudioContext фінгерпринтинг

Локальний час виводиться відповідно геолокації, але системний час зчитується і показується його фактичне значення (рис. 3.36)

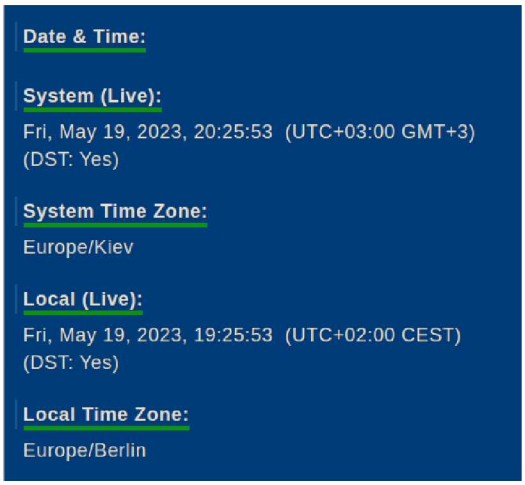

Рисунок 3.36 – Дата та час

Якщо використовувати проксі-сервер (-p) замість TOR (-t), то буде виводитись інформація про IP-адресу та геолокацію, що також не відповідає реальним даним користувача, але при цьому немає інформації про те, що використовується проксі-сервер (рис. 3.37 – 3.40). Локальний час також виводиться відповідно геолокації, але системний час зчитується і показується його фактичне значення (рис. 3.40)

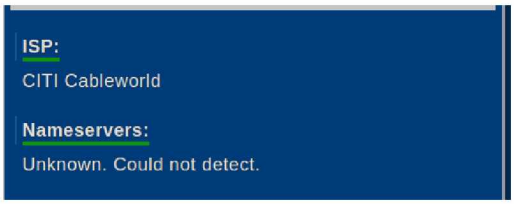

Рисунок 3.37 - Інформація про ISP

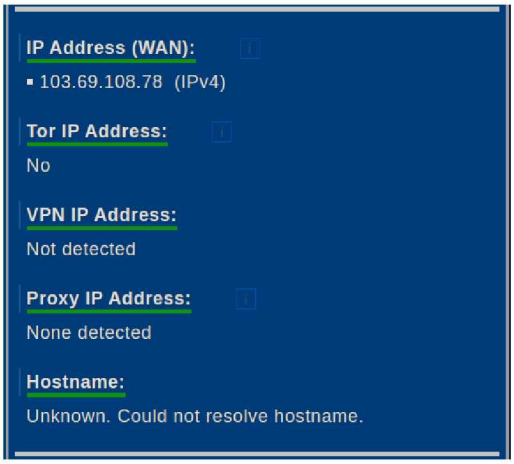

Рисунок 3.38– Інформація про IP-адресу (використання проксі)

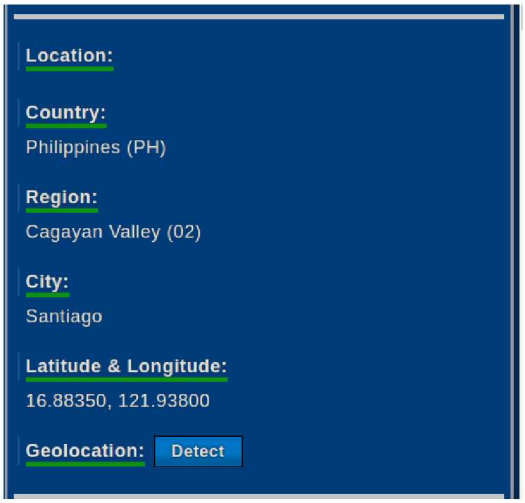

Рисунок 3.39 – Геолокація

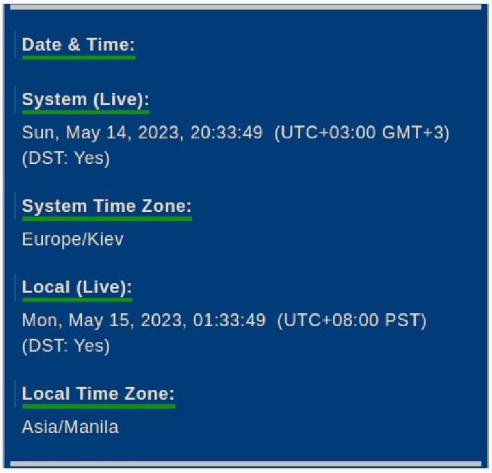

Рисунок 3.40 – Дата та час

Якщо використовувати обидва параметри TOR і проксі одночасно, то виходить створити ланцюг анонімності TOR+Проксі->Інтернет і приховати факт використання вузла мережі TOR (рис. 3.41).

Рисунок 3.41– Застосування TOR-адреси і проксі-сервера

Тож, ресурс DeviceInfo.me при тестуванні розробленого програмного рішення допоміг визначити, які параметри браузера були модифіковані, заблоковані або лишились незмінними та на практиці переконатися, що програма, яка підлягає тестуванню, виконує заявлені функції.

Сканування вебпереглядача з використанням сервісу Cover Your Tracks

Cover Your Tracks (coveryourtracks.eff.org) – це вебсайт, розроблений Electronic Frontier Foundation (EFF), некомерційною організацією, яка займається захистом цифрової конфіденційності користувачів мережі Інтернет. Вебсайт надає інструмент, який допомагає користувачам перевірити ефективність свого веббраузера щодо захисту від методів онлайн-відстеження. Інструмент сканує браузер і перевіряє наявність різних механізмів відстеження, таких як файли cookie, цифрові відбитки браузера тощо. Потім він надає вам детальний звіт про те, наскільки добре браузер захищає приватність користувача, і пропонує рекомендації щодо покращення налаштувань конфіденційності [11].

Тож виконаємо спроби сканування браузера Google Chrome при різних конфігураціях:

1. Спочатку зробимо спробу сканування веббраузера без застосування розробленої програми та будь-яких інших інструментів або розширень для захисту від вебвідстеження. На рис 3.42 наведений результат сканування браузера.

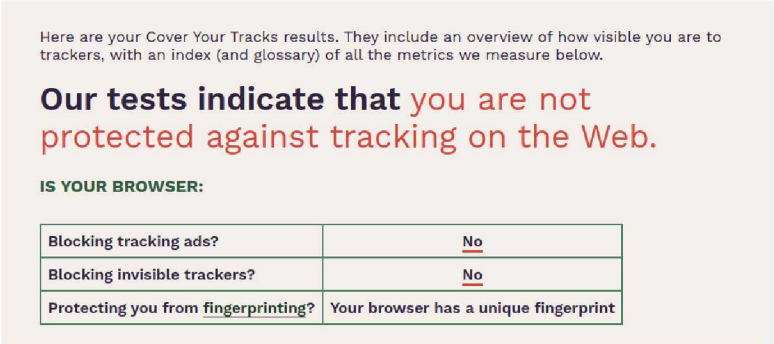

Рисунок 3.42 – Результат сканування браузера із налаштуваннями за замовчуванням

Отже, тестування CoverYourTracks показало, що якщо використовувати Google Chrome з налаштуваннями за замовчуванням, то користувач не буде захищеним в мережі Інтернет.

2. Запустивши ПЗ із аргументами *-p, -c, -i, -l, ua, wW, wH* (рис. 3.43), що виконають зміну IP-адреси з використанням проксі, відключення файлів cookie, застосують приватний режим перегляду, змінять набір мов інтерфейсу, агент користувача, розмір вікна браузера. Тож застосувавши зазначені конфігурації, отримаємо результат сканування, що наведений на рис. 3.44

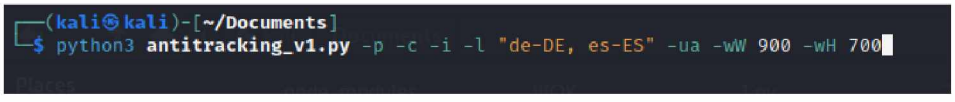

Рисунок 3.43– Запуск програми з агрументами *-p, -c, -i, -l, ua, wW, wH*

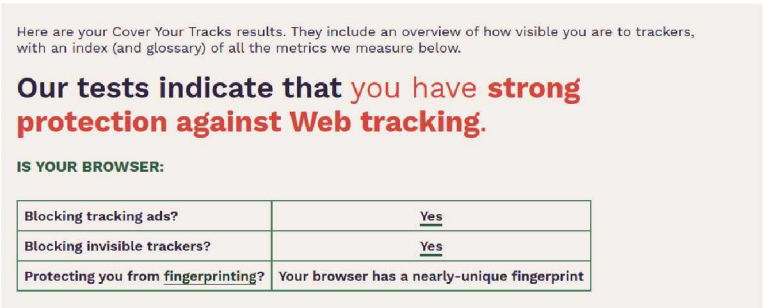

Рисунок 3.44 – Результат сканування браузера з модифікованими конфігураціями

3. Якщо запустимо програмний засіб із аргументами *-p, -js, -c, -i, -l, ua, wW, wH* (рис. 3.45), що змінять IP-адресу з використанням проксі, відключать javascript і файли cookie, застосують приватний режим перегляду, змінять набір мов інтерфейсу, агент користувача, розмір вікна браузера. Результат сканування браузера з новими конфігураціями наведений на рис. 3.46.

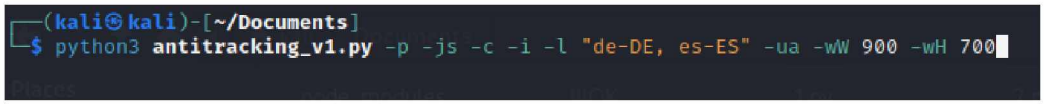

Рисунок 3.45 – Запуск програми з агрументами - *p, -js, -c, -i, -l, ua, wW, wH*

Рисунок 3.46 - Результат сканування браузера з модифікованими конфігураціями

4. Запустивши ПЗ із аргументами *-t, -c, -i, -l, ua, wW, wH* (рис. 3.47), IP-адреса буде змінена на IP-адресу вузла TOR, вимкнуться файли cookie, застосується приватний режим перегляду, зміниться набір мов інтерфейсу, агент користувача, розмір вікна браузера. Результат сканування браузера із зазначеними конфігураціями на рис. 3.48

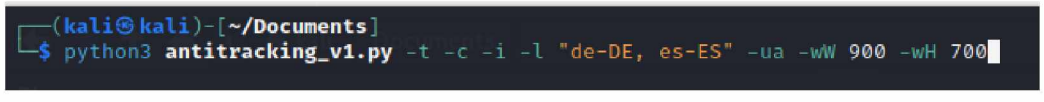

Рисунок 3.47– Запуск програми з агрументами *-t, -c, -i, -l, ua, wW, wH*

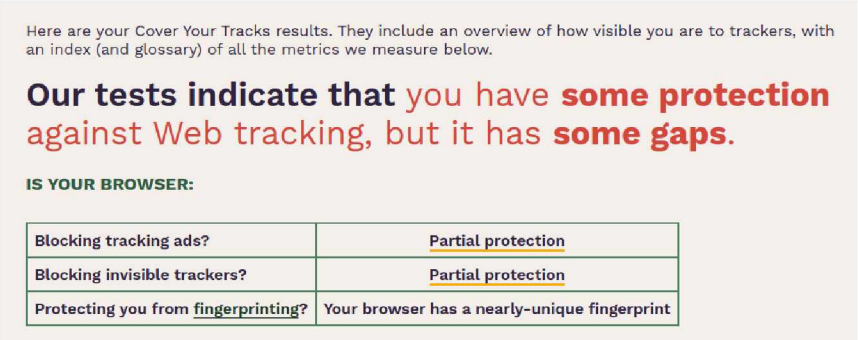

Рисунок 3.48- Результат сканування браузера з модифікованими конфігураціями

5. Якщо запустимо програмний засіб із параметрами *-t, -js, -c, -i, -l, ua, wW, wH* (рис. 3.49), то IP-адреса буде змінена на IP-адресу вузла TOR, відключиться javascript і файли cookie, застосується приватний режим перегляду, зміниться набір мов інтерфейсу, агент користувача, розмір вікна браузера. Отриманий результат сканування на рис. 3.50.

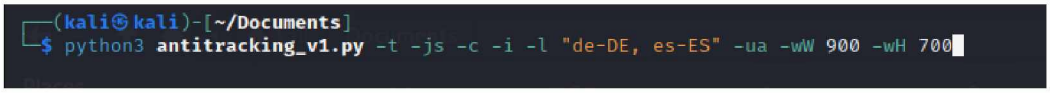

Рисунок 3.49– Запуск програми з агрументами *-t, -js, -c, -i, -l, ua, wW, wH*

Рисунок 3.50 – Результат сканування браузера з модифікованими конфігураціями

6. При запуску програми із аргументами командного рядка *-t, -p, -c, -i, -l, ua, wW, wH* (рис. 3.51) IP-адреса буде змінена із використанням ланцюга анонімності Проксі+TOR->Інтернет, вимкнуться файли cookie, застосується приватний режим перегляду, зміниться набір мов інтерфейсу, агент користувача, розмір вікна браузера. Тестування браузера при зазначених конфігураціях покаже результат, наведений на рис. 3.52

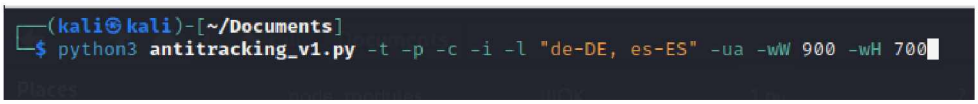

Рисунок 3.51– Запуск програми з агрументами *-t, -p, -c, -i, -l, ua, wW, wH*

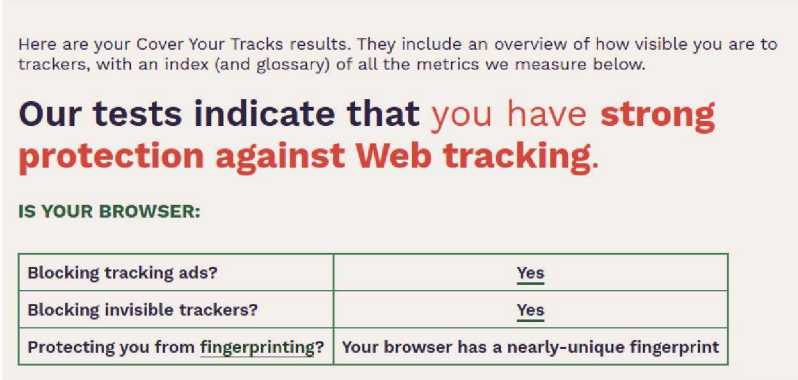

Рисунок 3.52 – Результат сканування браузера з модифікованими конфігураціями

Отже, виконавши тестування браузера при його запуску з різними конфігураціями, можемо прийти до висновку, що найефективнішим був захист при налаштуваннях, застосованих у пунктах 2 і 5 – сервіс CoverYourTracks під час сканування браузера виявив, що виконується блокування трекерів та частково рандомізується цифровий відбиток вебпереглядача. Тому можна вважати, що запуск програмного засобу із зазначеними аргументами надає комплексний захист від вебвідстеження.

Налаштування, що були застосовані у пунктах 3 та 4 також сервіс визначає як сильний захист від вебтрекінгу, але зазначається, що забезпечується захист тільки від трекерів, проте не від фінгерпринтингу – цифровий відбиток не рандомізується.

Середній рівень захисту показала конфігурація у пункті 4 – частково блокуються трекери і так само частково рандомізується цифровий відбиток браузера.

3.4 Переваги, недоліки та способи вдосконалення застосунку

Якщо оцінювати переваги розробленого програмного засобу, то до них можна віднести, по-перше, те, що він є повністю безкоштовним і може бути доступний для будь-якого користувача мережі Інтернет, який має на меті підвищити рівень своєї

анонімності та захистити свій вебпереглядач від відстеження різними вебсайтами та компаніями. По-друге, залежно від того, які параметри для захисту застосовувати, програмний засіб може слугувати комплексним рішенням і для блокування трекерів, і для захисту від фінгерпринтингу.

Недоліком даного програмного рішення є те, що у даній версії застосунку рандомізація цифрових відбитків браузера користувача не виконується цілком, а тільки частково. Проте, цей недолік не є критичним і він підлягає виправленню у наступній версії шляхом підміни значень більшої кількості параметрів браузера.

Можливим способом покращення розробленого застосунку є розширення його функціональних можливостей, щоб виконувати спуфінг більшої кількості параметрів браузера користувача. Наприклад, це можуть бути наступні параметри:

- тайм-зона;
- значення Canvas;
- значення WebGL;
- значення AudioContext;
- розширення браузера;
- встановлені шрифти;
- дані про апаратне забезпечення (оперативна пам’ять, процесор, стан батареї)

тощо.

Одним з варіантів покращення програмного засобу є інтегрування серверів VPN на додачу до проксі-серверів та мережі TOR для захисту IP-адреси користувача. Також для вдосконалення програмного засобу є варіант впровадити замість безкоштовних проксі застосовування платних рішень для більшої безпеки даних користувачів, щоб запобігти можливості витоку реальної IP-адреси через проміжний сервер. Для збільшення зручності використання додати до розробки графічний інтерфейс.

Висновки до розділу 3

У третьому розділі був програмно реалізований застосунок, призначений для підвищення рівня користувача в глобальній мережі шляхом спуфінгу цифрових

відбитків браузера. Наведений опис функцій програмного коду та пояснена логіка їх роботи. Також було проведене тестування розробленого програмного засобу і винесені наступні висновки:

1. Сканування браузера без використання жодних інструментів для захисту від вебвідстеження показує, що користувач взагалі не захищений в мережі Інтернет від відстеження. Натомість при використанні розробленої програми, сканування показує результати, що користувач або захищений частково, або має високий рівень захисту(відповідно до комбінації аргументів, які він застосовує).

2. Нерезультативно виконувати спуфінг тільки якогось одного певного параметра, використовуючи аргументи командного рядка при запуску програми поодинці, оскільки захист від фінгерпринтингу працює тільки якщо відбувається спуфінг низки параметрів браузера.

3. Щоб використання програмного засобу було ефективним, рекомендується використовувати комбінації налаштувань, наведених вище в пункті 3.3, залежно від потреб користувача.

В результаті тестування розробленого застосунку можемо дійти висновку, що дана програма виконує поставлену мету, яка полягає у підвищенні рівня анонімності користувача в глобальній мережі.

ВИСНОВКИ

Результатом написання кваліфікаційної роботи є розроблений програмний засіб, що дозволяє захистити користувачів від стеження в браузері, забезпечуючи підвищення рівня анонімності в мережі Інтернет.

Виконано аналіз нормативно-правової бази щодо вебтрекінгу в Україні та світі. Досліджено основні методи вебтрекінгу, такі як відстеження IP-адреси, файли cookie, фінгерпринтинг в браузері, пікселі відстеження. В ході дослідження виявлено, що найбільш інвазивним і складним для захисту з наведених методів вебвідстеження є фінгерпринтинг в браузері.

Проведено аналіз існуючих сучасних методів захисту проти вебтрекінгу. До них віднесено технології VPN, проксі-сервери та мережа TOR, вбудовані в браузерах функції захисту проти вебтрекінгу та програмні розширення, призначені для запобіганню фінгерпринтингу.

Розроблено алгоритм програмного засобу для запобігання відстеженню в мережі Інтернет шляхом зміни деяких конфігурацій браузера Chrome на випадкові або задані користувачем значення з метою приховування реальних його даних від вебсайтів і перешкоджання утворенню унікального ідентифікатора користувача. Виконано програмну реалізацію розробленого алгоритму.

Проведено тестування розробленого програмного засобу за допомогою декількох онлайн-ресурсів та показано його ефективність. Результати тестування ПЗ довели, що розроблений засіб виконує поставлені перед ним задачі – рівень захисту веббраузера користувача від вебтрекінгу зростає в залежності від застосованого функціоналу програми та рівень анонімності користувача підвищується за рахунок спуфінгу реальних конфігурацій, що застосовуються для генерування цифрового відбитку.

Запропоновано шляхи вдосконалення розробленої програми, в тому числі розширення його функціональних можливостей та інтегрування існуючих рішень для запобігання вебтрекінгу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mark Nottingham. Unsanctioned Web Tracking [Електронний ресурс] – Режим доступу: https://w3ctag.github.io/unsanctioned-tracking/

2. Загальний регламент про захист даних (GDPR) [Електронний ресурс]: Регламент Європейського Союзу від 25.05.2018 № 679. – Режим доступу: https://gdpr-info.eu

3. E-privacy Directive 2009/136/EC від 25.11.2009. [Електронний ресурс]: – Режим доступу: https://edps.europa.eu/data-protection/our-work/publications/legislation/directive-2009136ec_en

4. California Consumer Privacy Act (CCPA) від 10.05.2023. [Електронний ресурс]: – Режим доступу: https://oag.ca.gov/privacy/ccpa

5. Про захист персональних даних [Електронний ресурс]: Закон України від 29.07.2022 № 2494-IX. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2297-17#Text

6. Про рекламу [Електронний ресурс]: Закон України від 13.12.2022 № 2849-IX. – Режим доступу: https://zakon.rada.gov.ua/laws/show/270/96-вр#Text

7. Nikiforakis N., Kapravelos A., Joosen W., Kruegel, C., Piessens F., Vigna G. Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting. IEEE Symposium on Security and Privacy, 2013, pp. 541-555.

8. Gertjan Franken, Tom Van Goethem, Wouter Joosen. Who Left Open the Cookie Jar? A Comprehensive Evaluation of Third-Party Cookie Policies. 27th USENIX Conference on Security Symposium, 2018, pp. 151–168.

9. Pierre Laperdrix, Nataliia Bielova. Browser Fingerprinting: A survey. ACM Transactions on the Web, 2020, pp. 1-33.

10. Jonathan R. Mayer, John C. Mitchell. Third-Party Web Tracking: Policy and Technology. IEEE Symposium on Security and Privacy, 2012, pp. 413-427.

11. Cover Your Tracks [Електронний ресурс]: – Режим доступу: https://coveryourtracks.eff.org

12. A. Barth. [RFC6454] The Web Origin Concept. IETF, 2011. [Електронний ресурс] – Режим доступу: https://tools.ietf.org/html/rfc6454

13. Karsten Joost, Peter J. Hansen. The Web never forgets: Persistent tracking mechanisms in the wild. ACM SIGSAC Conference on Computer and Communications Security, 2014, pp. 674–689.

14. K. Mowery, H. Shacham. Pixel perfect: Fingerprinting canvas in HTML5. Web 2.0 Workshop on Security and Privacy (W2SP), IEEE Computer Science, 2012, 12 p.

15. Jordan S. Queiroz, Eduardo L. Feitosa. A Web Browser Fingerprinting Method Based on the Web Audio API. The Computer Journal, 2019, pp. 1106–1120.

16. R. Fielding, Ed.; J. Reschke. [RFC7231]Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content., Ed.. IETF, 2014. [Електронний ресурс] – Режим доступу: https://tools.ietf.org/html/rfc7231

17. Browser detection using the user agent [Електронний ресурс]: – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Browser_detection_using_the_user_agent

18. Fouad, Bielova, Legout, Sarafijanovic-Djukic. Missed by Filter Lists: Detecting Unknown Third-Party Trackers with Invisible Pixels [Електронний ресурс]: – Режим доступу:https://www.academia.edu/51475647/Missed_by_Filter_Lists_Detecting_Unknown_Third_Party_Trackers_with_Invisible_Pixels

19. Mandeep Pannu. Exploring Proxy Detection Methodology. The 4th International Conference on Cybercrime and Computer Forensics (ICCCF), 2016, 7 p.

20. M. Ligh, S. Adair, B. Hartstein and M. Richard. Malware Analyst's Cookbook and DVD : Tools and Techniques for Fighting Malicious Code. John Wiley & Sons, 2010, pp. 11-15.

21. E. Ramadhani. Anonymity communication VPN and Tor: a comparative study. Journal of Physics Conference Series, 2018, 6 p.

22. 25 million free VPN user records exposed [Електронний ресурс] – Режим доступу: https://cybernews.com/security/25-million-free-vpn-user-records-exposed/

23. David M. Goldschlag, Michael G. Reed, Paul F. Syverson. Onion Routing for Anonymous and Private Internet Connections. Communications of the ACM, 1999, pp 39–

24. TOR project. [Електронний ресурс] – Режим доступу: https://www.torproject.org

25. Dutta, R., Das, S., Chakraborty, S., Dasgupta, P. Improving Anonymity by Using a Mix of Proxies, VPNs and Onion Routing. International Journal of Computer Applications, 2017, pp. 21-26.

26. Englehardt, S., Narayanan, A. Online tracking: A 1-million-site measurement and analysis. ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 1388-1401.

27. Enhanced Tracking Protection in Firefox for desktop [Електронний ресурс] – Режим доступу: https://support.mozilla.org/en-US/kb/enhanced-tracking-protection-firefox-desktop

28. Firefox tracking protection [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/Privacy/Firefox_tracking_protection

29. Safari Privacy Overview [Електронний ресурс] – Режим доступу: https://www.apple.com/safari/docs/Safari_White_Paper_Nov_2019.pdf

30. Intelligent Tracking Prevention 2.3 [Електронний ресурс] – Режим доступу: https://webkit.org/blog/9521/intelligent-tracking-prevention-2-3/

31. Примітка про конфіденційність Google Chrome [Електронний ресурс] – Режим доступу: https://www.google.com/intl/uk/chrome/privacy/

32. Veronica Marotta, Vibhanshu Abhishek, Alessandro Acquisti. Online Tracking and Publishers' Revenues: An Empirical Analysis. Working Paper, University of Minnesota, Minneapolis, 2019, 35 p.

33. Trace Extension [Електронний ресурс] – Режим доступу: https://absolutedouble.co.uk/trace/

34. Nampoina Andriamilanto, Tristan Allard, Gaëtan Le Guelvouit, Alexandre Garel. A Large-scale Empirical Analysis of Browser Fingerprints Properties for Web Authentication. ACM Transactions on the Web, 2022, pp. 1-62.

35. Canvas Blocker (Fingerprint Protect) [Електронний ресурс] – Режим доступу: https://add0n.com/canvas-fingerprint-blocker.html

36. Privacy Badger Extension [Електронний ресурс] – Режим доступу: https://privacybadger.org

37. Baiju Muthukadan. Selenium with Python [Електронний ресурс] – Режим доступу: https://selenium-python.readthedocs.io

38. Will Keeling. Selenium-wire 5.1.0 [Електронний ресурс] – Режим доступу: https://pypi.org/project/selenium-wire/

39. Kenneth Reitz. Requests: HTTP for Humans [Електронний ресурс] – Режим доступу: https://requests.readthedocs.io/en/latest/

40. lxml - XML and HTML with Python [Електронний ресурс] – Режим доступу: https://lxml.de

41. Random — Generate pseudo-random numbers [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/random.html

42. Time – Time access and conversions [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/time.html

43. Argparse — Parser for command-line options, arguments and sub-commands [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/argparse.html

44. json — JSON encoder and decoder [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/json.html

45. PyAutoGUI's documentation [Електронний ресурс] – Режим доступу: https://pyautogui.readthedocs.io/en/latest/

46. Damian Johnson. Stem 1.8.2 [Електронний ресурс] – Режим доступу: https://pypi.org/project/stem/

47. subprocess — Subprocess management [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/subprocess.html

48. DeviceInfo.me [Електронний ресурс] – Режим доступу: https://www.deviceinfo.me

ДОДАТОК А
СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИПЛОМНОЇ РОБОТИ

Тези наукових доповідей

1. Кисельова А., Даков С., Дакова Л. Дослідження аспектів впливу на інформаційну безпеку особистості. V Міжнародна науково-практична конференція проблеми кібербезпеки інформаційно-телекомунікаційних систем (PCSITS). – Київ 2022. – С. 136

2. Кисельова А., Даков С. Метод збору інформації про користувача в мережі Інтернет з використанням цифрових відбитків браузера. VI Міжнародна науково-практична конференція проблеми кібербезпеки інформаційно-телекомунікаційних систем (PCSITS). – Київ 2023.

ДОДАТОК Б

Лістинг програмного засобу

Файл «antitracking_v1.py»

```
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from seleniumwire import webdriver
from stem import Signal
from stem.control import Controller
from lxml.html import fromstring
import requests
import random
import time
import argparse
import json
import pyautogui



def parse_arguments():
    parser = argparse.ArgumentParser(
        prog='Antifingerprinting module',
        description='***Program Functionality***',
        epilog='This program was created by (c)Kyselova Alina')
    parser.add_argument(
        '-js', '--javascript', action='store_true', help='disable JS')
    parser.add_argument(
        '-ua', '--useragent', action='store_true', help='switch to a random User-
Agent')
    parser.add_argument(
        '-wW', '--winWidth', action='store', help='customize window width')
    parser.add_argument(
        '-wH', '--winHeight', action='store', help='customize window height')
    parser.add_argument(
        '-l', '--languages', action='store',
        help='hide interface languages; argument must be passed in format "uk-UA, de-
DE, es-ES"')
    parser.add_argument(
        '-c', '--cookies', action='store_true', help='disable cookies')
    parser.add_argument(
        '-i', '--incognito', action='store_true', help='enable incognito mode')
    parser.add_argument(
        '-p', '--proxy', action='store_true', help='use Proxy to hide your IP')
    parser.add_argument(
        '-t', '--tor', action='store_true',
        help='use Tor to hide your IP; you must start Tor service before launch this
script')
    return parser.parse_args()


def get_proxies(r):
    url = 'https://free-proxy-list.net/'
    response = requests.get(url)
    parser = fromstring(response.text)
    proxies = set()
    for i in parser.xpath('//tbody/tr')[:r]:
        if i.xpath('.//td[7][contains(text(),"yes")]'):
```

```
            proxy = ":".join([i.xpath('.//td[1]/text()')[0],
i.xpath('.//td[2]/text()')[0]])
            proxies.add(proxy)
    return proxies



def check_proxies(proxies):
    valid_proxies = []
    for proxy in proxies:
        try:
            response = requests.get('https://www.google.com/', proxies={'http':
proxy, 'https': proxy}, timeout=10)
            if response.status_code == 200:
                valid_proxies.append(proxy)
            else:
                pass
        except:
            pass
    return valid_proxies



def read_data_json():
    with open('headers.json') as f:
        data = json.load(f)
    return data



def set_preferences(options, preferences, args):
    if args.tor:
        with Controller.from_port(port=9051) as controller:
            controller.authenticate(password='abc123')
            controller.signal(Signal.NEWNYM)
        options_s = {
            'proxy': {
                'http': 'socks5h://127.0.0.1:9050',
                'https': 'socks5h://127.0.0.1:9050',
                'connection_timeout': 10
            }
        }

    if args.proxy:
        i = 15
        proxies = []
        while i < 1000:
            if len(proxies) < 1:
                proxies = list(get_proxies(i))
                i = i + 15
            else:
                break
        valid_proxies = check_proxies(proxies)
        proxy = str(random.choice(valid_proxies))
        options_s = {
            'proxy': {
                'http': f'http://{proxy}',
                'https': f'https://{proxy}'
                # 'connection_timeout': 10
            }
        }
```

```
        preferences.update({"webrtc.ip_handling_policy": "disable_non_proxied_udp"})
        preferences.update({"webrtc.multiple_routes_enabled": False})
        preferences.update({"webrtc.nonproxied_udp_enabled": False})
        print(f"PROXY IP: {proxy}")

    if args.languages:
        preferences.update({"intl.accept_languages": f"{args.languages}"})
        print(f"INTERFACE LANGUAGES: {args.languages}")

    if args.cookies:
        preferences.update({"profile.default_content_settings.cookies": 2})
        preferences.update({"profile.block_third_party_cookies": True})
        preferences.update({"profile.default_content_setting_values.cookies": 2})
        print("COOKIES: disabled")

    if args.winWidth or args.winHeight:
        screen_width, screen_height = pyautogui.size()
        screen_width = int(screen_width)
        screen_height = int(screen_height)
        flag = True
        try:
            args.winWidth = int(args.winWidth)
            args.winHeight = int(args.winHeight)
        except:
            print("Value Error: You did not pass any int value!")
            flag = False
        if flag:
            if args.winWidth <= screen_width and args.winHeight <= screen_height:
                options.add_argument(f'--window-
size={args.winWidth},{args.winHeight}')
                print(f"WINDOW SIZE: {args.winWidth}x{args.winHeight}")
            elif args.winWidth and args.winWidth <= screen_width:
                options.add_argument(f'--window-
size={args.winWidth},{screen_height}')
                print(f"WINDOW SIZE: {args.winWidth}x{screen_height}")
            elif args.winHeight and args.winHeight <= screen_height:
                options.add_argument(f'--window-
size={screen_width},{args.winHeight}')
                print(f"WINDOW SIZE: {screen_width}x{args.winHeight}")

    if args.javascript:
        preferences.update({"profile.managed_default_content_settings.javascript":
2})
        print("JAVASCRIPT: disabled")

    if args.incognito:
        options.add_argument("--incognito")

    options.add_experimental_option("prefs", preferences)
    driver = webdriver.Chrome(options=options, seleniumwire_options=options_s)

    if args.useragent:
        # for change register of dictionary keys
        data = read_data_json()
        d = random.choice(data)
        d = {k.lower(): d[k] for k in d}
        user_agent = d['user-agent']

        def interceptor(request):
            del request.headers["user-agent"]
            for header in d.keys():
                request.headers[f"{header}"] = d[f"{header}"]
        driver.request_interceptor = interceptor
        print(f'USER AGENT: {user_agent}')
```

```
    return driver


def main():
    args = parse_arguments()
    url = "http://httpbin.org/anything"
    options = Options()
    # options = webdriver.ChromeOptions()
    # driver = webdriver.Chrome()
    preferences = {}

    print()
    print("========Welcome to Antifingerprinting App========")
    print()
    print("This program was created for protection against fingerprinting in Google
Chrome browser")
    print()
    print("========Your set parameters=====================")

    # options.add_argument("--disable-extensions")
    driver = set_preferences(options, preferences, args)
    driver.get(url)
    data = driver.find_element(By.TAG_NAME, "body").text

    try:
        data_json = json.loads(data)
    except:
        pass

    if args.tor:
        if 'origin' in data_json:
            print(f"TOR IP: {data_json['origin']}")

    print()
    print("================================================")
    print()
    flag = True
    time.sleep(10)
    while flag:
        flag = input("Are you want continue safe browsing(Y/n)?")
        if flag == 'Y' or flag == 'y':
            flag = True
            time.sleep(200)
        else:
            break

    # time.sleep(200)
    driver.quit()



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

Файл «headers.json»


```
[
    {
        "upgrade-insecure-requests": "1",
        "user-agent": "Mozilla/5.0 (Windows NT 10.0; Windows; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/103.0.5060.114 Safari/537.36",
        "accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
        "sec-ch-ua":    "\".Not/A)Brand\";v=\"99\",    \"Google    Chrome\";v=\"103\",
\"Chromium\";v=\"103\"",
        "sec-ch-ua-mobile": "?0",
        "sec-ch-ua-platform": "\"Windows\"",
        "sec-fetch-site": "none",
        "sec-fetch-mod": "",
        "sec-fetch-user": "?1",
        "accept-encoding": "gzip, deflate, br",
        "accept-language": "bg-BG,bg;q=0.9,en-US;q=0.8,en;q=0.7"
    },
    {
        "upgrade-insecure-requests": "1",
        "user-agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/103.0.5060.53 Safari/537.36",
        "accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
        "sec-ch-ua":    "\".Not/A)Brand\";v=\"99\",    \"Google    Chrome\";v=\"103\",
\"Chromium\";v=\"103\"",
        "sec-ch-ua-mobile": "?0",
        "sec-ch-ua-platform": "\"Linux\"",
        "sec-fetch-site": "none",
        "sec-fetch-mod": "",
        "sec-fetch-user": "?1",
        "accept-encoding": "gzip, deflate, br",
        "accept-language": "fr-CH,fr;q=0.9,en-US;q=0.8,en;q=0.7"
    },
    {
        "User-Agent":   "Mozilla/5.0   (Windows   NT   10.0;   Win64;   x64;   rv:98.0)
Gecko/20100101 Firefox/98.0",
        "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.
8",
        "Accept-Language": "en-US,en;q=0.5",
        "Accept-Encoding": "gzip, deflate",
        "Connection": "keep-alive",
        "Upgrade-Insecure-Requests": 1,
        "Sec-Fetch-Dest": "document",
        "Sec-Fetch-Mode": "navigate",
        "Sec-Fetch-Site": "none",
        "Sec-Fetch-User": "?1",
        "Cache-Control": "max-age=0"
    },
    {
        "Connection": "keep-alive",
        "Cache-Control": "max-age=0",
        "sec-ch-ua":  "\"  Not  A;Brand\";v=\"99\",  \"Chromium\";v=\"99\",  \"Google
Chrome\";v=\"99\"",
        "sec-ch-ua-mobile": "?0",
        "sec-ch-ua-platform": "macOS",
        "Upgrade-Insecure-Requests": 1,
        "User-Agent":   "Mozilla/5.0   (Macintosh;   Intel   Mac   OS   X   10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.83 Safari/537.36",
```

```
        "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
        "Sec-Fetch-Site": "none",
        "Sec-Fetch-Mode": "navigate",
        "Sec-Fetch-User": "?1",
        "Sec-Fetch-Dest": "document",
        "Accept-Encoding": "gzip, deflate, br",
        "Accept-Language": "en-GB,en-US;q=0.9,en;q=0.8"
    }
]
```