DOI 10.15276/imms.v16.no1.58
УДК 004.056.53

Informatics and Mathematical Methods in Simulation
Vol.16 (2026), No. 1, pp. 58-73

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ КРИМІНАЛІСТИЧНОГО АНАЛІЗУ ІСТОРІЇ ВЕББРАУЗЕРА

М.В. Відін[1], О.А. Стопакевич[1], А.О. Стопакевич[2]

[1]Національний університет «Одеська політехніка»
1, Шевченка пр., Одеса, 65044, Україна
[2]Державний університет інтелектуальних технологій та зв'язку
1, Кузнечна вул., Одеса, 65023, Україна
Email: stopakevich@gmail.com

Веббраузери є одним з основних джерел цифрових доказів в сучасній криміналістиці. Існуючі інструменти для криміналістичного аналізу профілів веббраузерів є недостатньо ефективними, оскільки відтворюють закладену в браузери модель даних, орієнтовану на швидкість рендерингу та оптимізацію роботи з окремими URL-адресами. Експерти-криміналісти змушені працювати з «сирими» списками, що містять тисячі записів, витрачаючи значний час на відтворення послідовності дій користувача, ручну фільтрацію технічних редиректів, трекерів, реклами та службових запитів. Мета роботи полягає у розробці алгоритмів та програмного застосунку для криміналістичного аналізу історії веббраузера Microsoft Edge, який трансформує розрізнені дані профілю у структуровану інформацію, придатну для оперативного виявлення інцидентів та реконструкції дій користувача. В основу розробки покладено перехід від класичної URL-орієнтованої моделі (характерної для структури баз даних Chromium) до хост-орієнтованої моделі представлення даних. Цей підхід дозволяє агрегувати тільки значимі артефакти активності (історію відвідувань, файли cookie, завантажені файли тощо) в хронологічному порядку навколо унікального імені хоста. Архітектурно рішення розділене на дві частини: модуль збору та агрегації даних, реалізований мовою Python з використанням асинхронних запитів для швидкої обробки масивів інформації, та модуль візуалізації на базі бібліотеки Webix, що забезпечує високу продуктивність інтерфейсу при роботі з великими даними. Графічний інтерфейс застосунку дозволяє експерту проводити багатопріоритетне сортування та фільтрацію записів за багатьма критеріями. Особливу увагу приділено візуалізації ланцюжків переходів, що дає змогу відтворити послідовність дій підозрюваного на конкретному ресурсі. Тестування підтвердило здатність застосунку швидко обробляти дані та виявляти релевантні докази, значно скорочуючи час, необхідний для експертизи, порівняно зі звичайним ручним аналізом «сирих» даних. Він забезпечує наочне представлення цифрових доказів та мінімізує ймовірність пропуску важливої інформації під час розслідувань.

Ключові слова: криміналістика; аналіз; браузер; історія; профіль; агрегація; застосунок; хост-орієнтована модель даних; SQLite; JSON; Python; JavaScript; Webix.

Вступ. Дослідження в галузі цифрової криміналістики веббраузерів переважно зосереджуються на аналізі та відновленні збережених в них даних з метою ідентифікації та збору важливих доказів щодо онлайн-поведінки об'єкта розслідування. Практично всі дії, які здійснює підозрюваний під час використання веббраузера, можуть залишити сліди на його комп'ютері. Тому вивчення цих доказів на пристрої підозрюваного може надати цінну інформацію для слідчих. Вилучення таких даних, як історія, файли cookie, списки завантажень, кеш, збережені паролі з веббраузера проводиться за допомогою спеціальних програмних засобів, огляд найбільш поширених з яких приведено в [1].

Зараз експерти-криміналісти мають вручну аналізувати велику кількість вебадрес (URL адрес), які зберігаються в профілі браузера. Їх доволі важко аналізувати вручну, оскільки типовий користувач відвідує від 100 до 500 сайтів на місяць в залежності від його сфери професійної діяльності, а кількість URL, звісно, буде в рази більшою.

Ручний аналіз історії браузера за місяць активності може займати у експерта від 4 до 8 годин робочого часу. Існуючі інструменти часто видають технічні таблиці, які містять перелік URL в хронологічному порядку. Для роботи з деякими програмами потрібні знання про структуру протоколів, вебпрограмування, будову браузерів тощо для правильної інтерпретації. Значний час уходить на фільтрацію сміттєвих вебадрес, реклами, трекерів, технічних редіректів тощо.

Основна причина полягає в тому, що внутрішній формат зберігання даних в браузерах розроблено виходячи з критеріїв швидкості виконання певних функцій. Основною одиницею даних є URL. Це зручно для браузера, наприклад, щоб виділити покликання, які були вже відвідані чи продовжити URL при введені його початку в адресний рядок. Проте не зручно для користувача який отримує перелік URL, які відкривав браузер, без систематизації й з сортуванням за датою останнього виклику. Бази даних профілю з метою пришвидшення не застосовують механізми забезпечення цілісності – ключі, вбудовані процедури, перевірки тощо. При цьому докладна інформація для оперативного доступу зберігається тільки для URL, які відвідувались за останні 3 місяця. Інформація про URL, останній доступ яких був понад три місяця, стирається з основних таблиць, але залишається в другорядних. Оскільки формат зберігання орієнтований на ефективність браузера, то єдиним шляхом зробити її ручним для людини є спеціальна обробка даних, агрегація зі зміною основної одиниці, що наявні утиліти не роблять.

Як універсальні програми для збору доказів, для яких аналіз даних профілів – лише одна з функцій, так й спеціалізовані утиліти для цієї задачі, у цілому недостатньо зручні. Вони просто представляють дані, виходячи зі схем, в яких вони зберігаються в профілі браузера. Як правило як перші, так й другі видають "сирі" списки URL, лишаючи інтерпретацію людині. При цьому ряд утиліт навіть не мають графічного інтерфейсу й призначені щоб експортувати дані в Excel, XML, HTML тощо.

Ця робота присвячена розробці застосунку, який дозволяє аналізувати активність користувача браузера більш зручним чином, ніж це пропонується в наявних програмних застосунках. Це дозволить оптимізувати процес: зменшити кількість часу на аналіз й скоротити ймовірність пропуску важливих даних.

Мета та задачі дослідження. Мета дослідження – розробити необхідні алгоритми та зробити принципові програмні рішення, які пов'язані з перетворенням інформації, яка зберігається в профілі веббраузера у інформацію, яка за схемою даних та видом представлення є більш структурованою та придатною для криміналістичного аналізу. Як веббраузер виберемо Microsoft Edge під ОС Microsoft Windows.

Задачі дослідження.

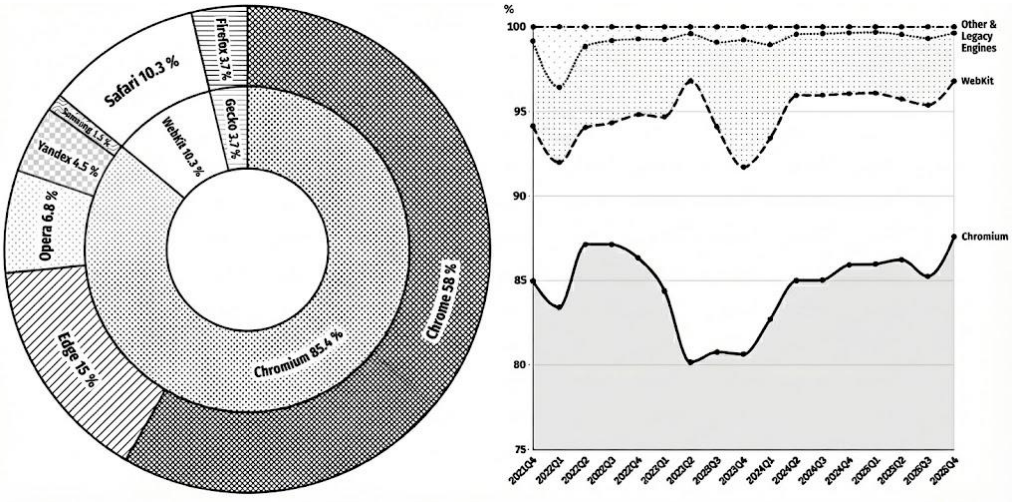
1. Короткий огляд архітектури сучасних браузерів на платформі Chromium.
2. Короткий огляд принципів роботи та функціональних можливостей наявних застосунків для аналізу профілів користувачів.
3. Формулювання переліку вимог до нового застосунку.
4. Вибір необхідних інструментів та бібліотек для виконання задачі розробки застосунку.
5. Розробка частини збору даних застосунку.
6. Розробка частини візуалізації застосунку.
7. Тестування застосунку на реальному профілі веббраузера.

Архітектура сучасних браузерів на платформі Chromium. Браузери на базі платформі Chromium найбільш популярні світі. Розподіл користувачів браузерів на початок 2025 р. й за останні 4 роки проілюстрований на рис. 1.

Якщо аналізувати певні тенденції, то можемо виявити наступні: доля браузерів на платформі Chromium ~ 85% й зростає; лідирує Chrome, Edge, Safari; втрачає позиції Firefox. Користувачі Safari – це користувачі пристроїв Apple, а користувачі Firefox – це

значною мірою користувачі Linux, деякі ІТівці й ті, хто прагне бути незалежними від великих корпорацій.

Технічно всі браузери на платформі Chromium мало відрізняються один від одного [2]. Все, що стосується основи — рендерингу, історії, кукі (cookies), обраного, внутрішніх настройок в них ідентично. Проєкт Chromium є формально незалежним й розробляє відкритий програмний код, який розробники браузерів (Google, Microsoft, Opera Software тощо) на базі цієї платформи доповнюють своїми патчами й додатковими модулями. Розробники браузерів додають різноманітний функціонал, наприклад інтеграцію двигуна ШІ, прив'язки акаунту до певних сервісів розробника браузера, синхронізацію даних між браузерами одного користувача на різних пристроях, підтримку VPN/оптимізатора/фільтру трафіку, підвищення рівня захисту, ефективності використання ресурсів тощо.



а) DataReportal 2025 р. б) StatCounter 2021-25 рр.

Рис.1. Розподіл між браузерами серед користувачів України

Браузер Edge має певні переваги, які проявляються переважно тільки в ОС Windows: тісна інтеграція з Windows, Office, Microsoft 365, економія батареї, економія ресурсів ОС, SmartScreen, ізоляція вкладок (більш безпечний механізм, ніж Sandbox реалізований в Chrome), Copilot як ШІ-помічник, найбільша кількість функціональних можливостей графічного інтерфейсу в порівнянні з аналогами. Достатньо об'єктивне та повне порівняння Edge та Chrome може бути переглянуте за [3]. Огляд можливостей інших браузерів на платформі Chromium приведений в [4].

У цілому в профілі браузерів на платформі Chromium основна інформація про відвідування зберігається в форматі баз даних SQLLite, а про настройки браузерів – в JSON. Частина даних (локальне сховище сайтів тощо) зберігається в інших форматах, наприклад IndexedDB. Схема зберігання даних кожного браузера зазвичай додає певні параметри й дані, але не видаляє й не змінює сенс тих, які реалізуються в вихідному коді платформи Chromium. Детальний огляд схеми даних браузера Edge, проведений нами в [1].

Принципи роботи та функціональні можливості наявних застосунків для аналізу профілів користувачів. Проблема безпеки даних, які зберігаються в даних, розглянута в [5]. Автори показують, що майже всі популярні браузери (крім Tor) зберігають критично важливі дані у домашніх директоріях без особливого захисту. Механізми шифрування паролів і cookies можна обійти, а HTTPS – зламати шляхом ін’єкції шкідливих кореневих сертифікатів. Таким чином, якщо є доступ до акаунту Windows користувача, отримати весь зміст профілю з домашньої папки не є складною задачею. Значною мірою

застосування стандартних підходів до зберігання й шифрування даних в профілі пояснюється тим, що вихідний код браузерів на базі платформ Chromium, WebKit, Gecko є загальнодоступним. Велика кількість можливостей сучасних браузерів, які застосовуються не тільки для сайтів, але й для реалізації застосунків з графічним інтерфейсів на десктопі та мобільних пристроях, призводить до потенційних вразливостей. В зазначеній роботі приведені приклади кібератак, які їх можуть експлуатувати.

Вилучати дані з профілю браузера можна 4 основними типами інструментів:

– спеціальні утиліти (в тому числі написані власноруч) , які збирають дані й записують їх в один з зручних форматів (Excel, HTML звіт тощо);

– утиліти для доступу до даних різних форматів (графічне середовище для роботи з БД SQLite, спеціалізовані редактори JSON, аналізатори дампу пам'яті тощо);

– комплексні застосунки для збору криміналістичних даних з ОС та різних програм, встановлених на диску користувача;

– спеціальні утиліти з графічним інтерфейсом, які призначені для роботи з профілями (повністю чи з конкретним аспектом) конкретних браузерів.

Огляд на наш погляд найбільш зручних утиліт трьох типів для браузерів на платформі Chromium проведений як в роботі [1], так і в роботах [6-8]. В цих роботах розглянуті програми для комплексного збору доказів й інші спеціальні утиліти, виділені основні їх можливості та зроблений порівняльний аналіз.

Формулювання переліку вимог до нового застосунку. Наша мета – розробити новий застосунок для роботи з профілями браузера Edge під ОС Windows, в якому за базу класифікації інформації з якою працює користувач буде взято ім'я хоста. Тому перед видаванням інформації в графічний інтерфейс користувача необхідно провести попередню агрегацію: сформувати на базі розкиданих даних цілісну базу даних для аналізу, яка буде зручною для перегляду та роботи. Застосунок має відповідати наступному переліку вимог:

– в основному орієнтація на ОС Windows, проте адаптація для других популярних ОС має бути тривіальною;

– інтерфейс має бути зрозумілим будь-якому IT-спеціалісту, неочевидні питання мають бути розкриті в довідці, яка доступна безпосередньо в інтерфейсі;

– основна мова – українська, якщо застосовуються невідомі англомовні терміни/скорочення, то їх пояснення українською мовою має реалізуватись через підказку при наведенні (tooltip);

– можливість швидкої роботи з даними обсягом до 50 тис. хостів й 500 тис. URL;

– результатом парсингу БД та JSON файлів профіля є JSON файл з агрегованими даними;

– частина візуалізації має виконуватись на Desktop й дозволяти відмічати цікаві та нецікаві записи й зберігати помітки при перезавантаженні сторінки/браузера;

– частина візуалізації має забезпечити швидку та зручну роботу з даними, що передбачає реалізацію функцій сортування (в тому числі багатопріоритетне) й фільтрування даних;

– частина візуалізації має мати адаптивний інтерфейс, який враховує можливості екрана в межах розмірів Desktop (мобільна версія чи версія для планшетів не передбачається);

– частина візуалізації має складатись з 4 вкладок: "Відвідування" (таблиця хостів + інформації про сесії відвідування їх URL за запитом), "Пошукові запити" (запити пошукових систем), "Про користувача" (інформація про акаунт, паролі, мовні вподобання), "Профіль" (контрольні суми, розмір, статистична інформація про файли профілю).

Вибір необхідних інструментів та бібліотек для виконання задачі розробки застосунку. В основу застосунку покладена хост-орієнтована модель представлення

даних. На відміну від оригінальної, URL-орієнтованої структури Chromium, ця модель агрегує всі розрізнені артефакти (історію, файли cookie, дані форм, кеш) навколо імені хоста. Це значно підвищує зручність, швидкість та глибину аналізу, дозволяючи аналітику-криміналісту отримати повну картину взаємодії користувача з кожним сайтом. Таким чином, застосунок розділяється на дві частини: частини збору та агрегування інформації та частину візуалізації з функціями пошуку, яка працює не з даними профілю, а з агрегованою інформацією.

Застосунок розділимо на дві окремі частини, які будуть реалізовуватись різними мовами програмування: частину агрегації та частину графічного інтерфейсу.

Перша частина застосунку реалізує набір алгоритмів для вилучення, очищення, перетворення та агрегації даних з профілю браузера у описаний специфікований формат JSON. Модуль враховує реальні проблеми цілісності даних, усуває неінформативний «шум» (локальні адреси, дані розширення) та додає інформацію, пов'язану з геолокацією хоста. Для реалізації першої частини застосунку будемо використовувати дистрибутив Anaconda на базі Python 3.12.

Серед стандартних бібліотек будемо використовувати: asyncio для ефективної роботи з сотнями одночасних мережевих з'єднань, datetime, ipaddress для перевірки зони (локальна, глобальна) IPv4, json, os, re, socket, sqlite3, typing, urllib для розбиття URL на компоненти (схема, хост, порт, шлях, параметри) й для нормалізації та перевірки їх коректності.

Серед нестандартних бібліотек будемо використовувати: Cipher – для вилучення паролів з профілю, ifaddr для визначення переліку IP-адрес мережевих карт, задіяних ОС, maxminddb для роботи з локальними геолокаційними базами даних MaxMind, pythonping для реалізації ping-запитів з можливістю настройки припустимого часу та інших параметрів, simple_cache для кешування результатів виконання витратних за часом функцій (мережевих запитів), для виклику криптографічних функцій Windows з метою дешифрації паролю.

Друга частина реалізує вебінтерфейс для візуалізації та аналізу отриманих даних. Для реалізації вебінтерфейсу будемо застосовувати мову JavaScript з комплексною бібліотекою компонентів, оскільки нам потрібно реалізовувати інтерфейс десктоп-застосунку. Основним компонентом є таблиця даних, зміст якої має генеруватись динамічно при прокрутці з метою підвищення ефективності роботи. Прокрутка, фільтрація, сортування з даними обсягами в десятки тисяч записів має бути миттєвою. У цілому критеріями для вибору бібліотеки графічних компонентів будуть:

- наявність ключових компонентів та їх функціональні можливості;
- продуктивність таблиць: віртуалізація, ефективність застосування DOM браузера, підтримка великих даних й рівень масштабованості;
- потрібний функціонал має бути безкоштовним й не ставити вимоги до програмного коду, який використовує бібліотеку;
- розмір бібліотеки;
- наявність додаткових інструментів, які спрощують розробку;
- можливості стилізації;
- якість документації.

Проведений порівняльний аналіз ряду бібліотек (Webix, Kendo UI, Ext JS, DevExtreme тощо) зупинив наш вибір на Webix. Згідно з даними дослідження [9] datatable цієї бібліотеки забезпечує порівняний з іншими рішеннями рівень продуктивності й відповідає вимогам масштабованості. Теж саме підтверджується в експериментальному дослідженні українських вчених [10]. В цілому Webix, безкоштовна версія, має достатню для наших задач функціональність.

Розробка частини збору даних застосунку. Схема даних для збору інформації, орієнтована на хост проілюстрована на UML-діаграмі класів (рис. 2).

Заповнення цієї схеми виконується згідно з розробленим алгоритмом.

Алгоритм виходить з визначення глобального IP, коректного хоста та коректного url.

Глобальний IP – це IP, який: 1) є коректною IPv4 адресою; 2) не є IP адресою адаптерів внутрішньої мережі; 3) не входить до переліку внутрішніх адрес, на кшталт 0.0.0.0, 127.0.0.1, 192.168.. тощо.

Коректний хост – це хост, який: 1) якщо містить IP, то це глобальний IP; 2) якщо це ім'я, то воно посилається на глобальний IP; 3) не містить ідентифікатор модуля Edge, тобто не відповідає "^[a-z]{32}$"; 4) не є внутрішньою сторінкою Edge, тобто не входить до переліку: "about", "edge-urls", "accessibility", "apps", "app-service-internals" тощо.

Коректний URL – це URL, який: 1) починається зі схеми http / https / ftp; 2) містить ім'я хоста чи ipv4 адресу й цей хост коректний; 3) у цілому відповідає структурі RFC 3986.

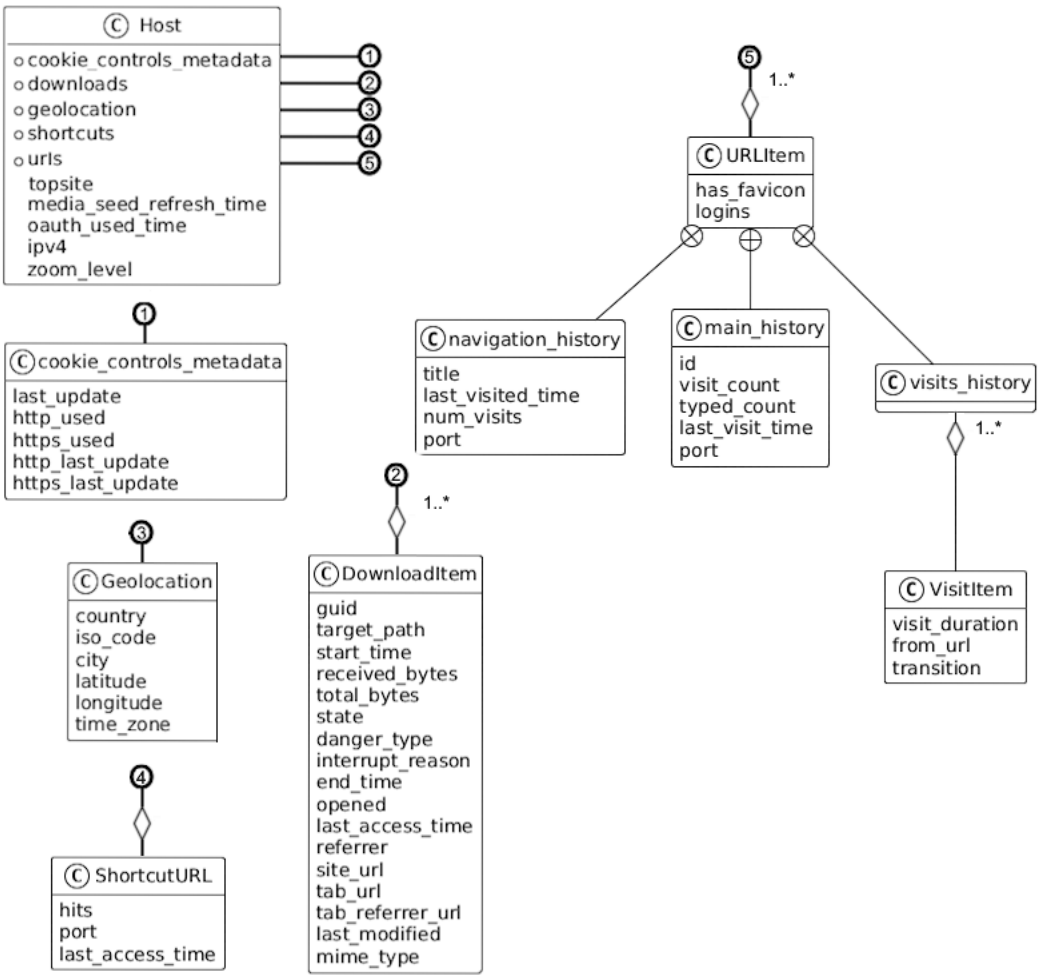


Рис. 2. UML діаграма схеми даних, яку заповнює парсер

Кроки виконання алгоритму.

К1. Збір усіх унікальних URL адрес з баз даних. Для цього застосовується SQL команда SELECT DISTINCT. Вибірка проводиться з webassist.navigation_history.url, history.urls.url, favicons.icon_mapping.page_url, topsites.top_sites.url.

К2. Виділення коректних URL та хостів. Для цього виконується.

1. Формування об'єднаного переліку всіх URL all_url_list з видаленням дублікатів.
2. Розділення переліку на перелік коректних та некоректних URL.
3. Виділення з переліку коректних URL переліку унікальних хостів з портами.
4. Розділення переліку на перелік коректних та некоректних хостів з портами.
5. Видалення хостів відомих трекерів, рекламних систем тощо.

6. Видалення локальних IP адрес та хостів, з ними пов'язаних. IP адреси збираються з кожного мережевого пристрою з використанням функцій бібліотеки ifaddr. Далі вони перевіряються на доступність за допомогою ping (бібліотека pythonping). Останній етап - отримуються хости, пов'язані з IP за допомогою socket.gethostbyaddr(ipaddr).

К3. Заповнення словника search_queries_dict переліком запитів до пошукових систем (Google, Bing тощо). Запити отримуються шляхом декодування URL, в яких записані пошукові запити в форматі RFC 1738. Всі додані в словник URL видаляються з зібраних первинних даних.

К4. Заповнення словника site_dict первинними даними відносно хостів (за рис. 2)

К5. Завантаження з файлу JSON preferences інформації про останні оновлення кукі відносно наявних хостів (з врахуванням протоколу). Конкретні значення кукі не завантажуються. Це нам дозволяє отримати дату останнього відвідування хоста.

К6. Завантаження з файлу JSON preferences збережених режимів масштабування для кожного хоста. Збереження величини проводиться в поле zoom_level запису про хост в словнику. Це непрямий вказівник на те, що інформація користувачем уважно читалась.

К7. Завантаження з файлу JSON preferences часу авторизації та URL з якого була проведена авторизація за механізмом OAUTH. Збереження часу проводиться в поле oauth_used_time запису про хост в словнику site_dict.

К8. Завантаження з таблиці БД topsites.top_sites переліку найбільш відвідуваних сайтів. Збереження результату проводиться в topsite запису про хост в словнику site_dict.

К9. Завантаження з БД history.downloads переліку збережених завантажень. Збереження результату проводиться в поле downloads запису про хост в словнику site_dict.

К10. Завантаження з БД media_device.media_device_salts переліку хостів, яким дозволено доступ до мультимедіа пристроїв. При обробці треба звернути увагу, що в storage_key може бути присутньо одночасно декілька хостів через ^0. Наприклад, це може бути https://linguacoach.appspot.com/^0https://pronunciationchecker.com. Збереження дати останньої генерації солі проводиться в поле media_seed_refresh_time запису про хост в словнику.

К11. Отримання для всіх хостів з доменними іменами IP адрес з використанням стратегії асинхронного отримання IP-адрес для списку доменних імен з обмеженням швидкості запитів, паралельним виконанням та кешуванням результатів.

Отримання IP адрес проводиться наступним чином:

1. Ініціалізація об'єкта обмеження швидкості (AsyncRateLimiter):
   - приймає регламент роботи "Token Bucket";
   - запускає фоновий цикл поповнення токенів (_refill_loop) через asyncio.create_task();
   - поповнення відбувається кожні 0.1 сек з інкрементом rate_per_sec * 0.1;
   - метод acquire() блокує виконання до отримання токена через цикл перевірки з asyncio.sleep(0.005).
2. Підготовка даних у resolve_hosts_ipv4_async:
   - уніфікація списку хостів зі збереженням порядку через dict.fromkeys();
   - розділення на кешовані/некешовані хости з використанням глобального словника dns_cache;
   - ініціалізація DNS-резолвера бібліотеки dns
     ```
     resolver = dns.asyncresolver.Resolver(configure=False)
     resolver.nameservers = [dns_server]
     ```
3. Паралельне встановлення відповідності (резолвінг):
   - створення семафора asyncio.Semaphore(concurrency) для обмеження одночасних запитів;
   - запуск асинхронних worker'ів через asyncio.create_task() для кожного некешованого хоста;

- використання asyncio.gather() для очікування завершення всіх задач.

4. Обробка результатів:
   - для IP-адрес прямим записом у результати (резолвінг не потрібен), дані мають фільтруватись на вході, це для безпеки;
   - для доменів – виклик_resolve_one з очікуванням токена limiter.acquire() та асинхронним запитом до DNS через resolver.resolve(host, 'A').
5. Збереження результатів у кеш за допомогою simple_cache.save_key().
6. Завершення роботи:
   - зупинка обмежувача через limiter.close() з відміною задачі_refill_task.cancel();
   - об'єднання кешованих і нових результатів.

В результаті отримаємо словник формату {хост: IPv4-адреса}.

К12. Отримання інформації щодо геолокації хостів з використанням локальної БД. При наявності нової версії БД програма має її завантажувати. Запити мають кешуватись. Збереження результату проводиться в поле geolocation запису про хост в словнику site_dict.

К13. Заповнення полів navigation_history, main_history, favicon за наявності інформації про відповідний запис URL, який створюється в запису відповідного хоста в словнику site_dict.

К14. Завантаження з БД history переліку відвідувань, який записується за наявності інформації в відповідний запис URL, який створюється в запису відповідного хоста. Необхідний SQL-запит наступний:

```
SELECT
u.url         AS url,
v.visit_time  AS visit_time,
printf('%d:%02d:%06.3f',
    cast((v.visit_duration/1000000) / 3600 as integer),
    cast(((v.visit_duration/1000000) % 3600) / 60 as integer),
    (v.visit_duration/1000000.0) % 60.0
) AS visit_duration,
COALESCE(u_from.url, '')  AS from_url,
v.transition as transition
FROM visits v
JOIN urls u ON u.id = v.url
LEFT JOIN visits v_from ON v_from.id = v.from_visit
LEFT JOIN urls u_from ON u_from.id = v_from.url
ORDER BY u.url, v.visit_time DESC;
```

К15. Завантаження з БД shortcuts лічильників ручного введення адреси, який записується за наявності інформації в відповідний запис URL, який створюється в запису відповідного хоста. Необхідний SQL-запит наступний:

```
SELECT url, SUM(number_of_hits) as noh, max(last_access_time) as lat
FROM omni_box_shortcuts WHERE keyword=""  GROUP BY url ORDER BY noh
DESC
```

К16. Завантаження з БД logindata переліку збережених паролів, якщо дослідження проводиться з використанням облікового запису користувача.

Спочатку завантажуємо ключ з файлу Local State

```
with open(storage_path + "Local State", 'r') as file:
    enc_key = json.loads(file.read())['os_crypt']['encrypted_key']
enc_key = base64.b64decode(enc_key)
enc_key = enc_key[5:]
dec_key = win32crypt.CryptUnprotectData(enc_key, None, None, None,
0)[1]
```

Далі отримуємо пароль для потрібного значення поля password_value таблиці logins

```
cipher = AES.new(dec_key, AES.MODE_GCM, nonce=enc_pwd[3:3+12])
```

```
dec_password=cipher.decrypt_and_verify(enc_pwd[3+12:-16], enc_pwd[-16:])
```

К17. Створення словника user_data_dict.

К18. Запис в словник інформації про файли досліджуваного профілю (дата, розмір, контрольна сума).

К19. Завантаження з файла за json_preferences_path інформації про власника акаунта Microsoft, який підключений до Edge.

К20. Завантаження з JSON файла preferences інформації про мовні вподобання користувача.

К21. Завантаження з JSON файлу preferences інформації про закладки користувача.

К22. Завантаження з БД shortcuts інформації про пошукові запити користувача. SQL-запит має наступний вигляд:

```
SELECT url, contents, keyword, SUM(number_of_hits) as noh, max(last_access_time) as lat
FROM omni_box_shortcuts WHERE keyword!="" GROUP BY url ORDER BY noh DESC
```

В БД зберігаються лише дані за 3 місяця. Але цей перелік можна об'єднати з інформацією, яка записана в search_queries_dict за весь період профілю (вилучити запити з URL хостів пошукових систем, які не мають обмежений термін збереження).

К23. Агрегація даних зі словників таким чином, щоб записати інформацію про відвідування URL завантаження в хостах в хронологічному порядку за сесіями з розшифровкою кодів transition (причина переходу).

К24. Запис агрегованих даних в JSON файл для подальшої роботи через графічний інтерфейс.

Розробка частини візуалізації застосунку. Графічний інтерфейс застосунку має бути зручним робочим інструментом для криміналіста та спеціаліста з кібербезпеки, який виконує низку задач, які необхідні для дослідження. Основна мета застосунку - візуалізація даних, кількість яких може бути доволі значною. Застосунок має виконуватись на ПК (Desktop).

Серед великої кількості відвіданих URL цікавими можуть виявитись доволі небагато. Якщо криміналіст знає що шукати (час, сайти), то система фільтрів має допомогти якнайшвидше отримати позитивний результат. Якщо ж ні, то система фільтрів має дозволити не заблукати в безлічі непотрібної інформації. Застосунок не має передбачати виконання якісь інтелектуальних операцій, його задача - знайти цікаві записи вручну. Подальший їх аналіз – це вже справа людини-спеціаліста.

Деталізуємо зміст 4 вкладок інтерфейсу, перелік яких зазначений в вимогах.

Вкладка "Відвідування" має містити:

1. таблицю з інформацією про хости, яка має:
   - поля: хост, статус, дата останнього відвідування (час відображається в підказці при наведенні), кількість візитів, локація фізичної точки доступу (Access Point), IPv4 адреса, кількість завантажень, наявність в топ 20 (признак), коефіцієнт масштабування, наявність авторизації (признак), наявність авторизації через OAUTH (признак), дозвіл на застосування мультимедіа пристроїв (признак), введення з адресного рядка хоча би в одному випадку (признак);
   - багатопріоритетне сортування з врахуванням типу даних, яке за замовчуванням ведеться по даті останнього відвідування в зворотному порядку;
   - відображати всі дані без розбивання на сторінки;
   - дозволяти отримати інформацію про IP адресу/Геолокацію за допомогою наявних сервісів шляхом клацання по відповідній комірці запису;

- можливість встановлювати позначку для кожного запису прямо в таблиці: цікаво, не цікаво, не дивився;
- можливість відкривати інформацію про відвідування конкретного хоста

2. фільтри для таблиці з інформацією про хости, які розділяються на:
   - бінарні (показувати/не показувати): цікаві, переглянуті, не переглянуті, поза TOP 20, без логіну (тобто такі, на які користувач не авторизувався);
   - інтервальні (дата з, дата по);
   - текстові з використанням регулярних виразів (хост, заголовок, IP);
   - інші (країна фізичної точки доступу хоста тощо).
3. деревовидну таблицю з інформацією про відвідування URL з хоста, яка:
   - заповнюється коли обирається конкретний хост в таблиці (клацають по його імені);
   - візуалізує ієрархію виду сайт -> ланцюжок -> url;
   - є меншою за висотою таблиці з інформацією про хости, проте може бути за потреби розширена на весь екран;
   - відображає за допомогою дерев інформацію, відсортовану за ланцюжками відвідувань в зворотному хронологічному порядку;
   - в кожному ланцюжку відображає шлях URL, заголовок сторінки, додаткову інформацію з поясненнями, час відвідування кожного URL та фінальний час відвідування ланцюжка;
   - дозволяти користувачеві перейти на певні сторінки клацанням, але з попередженням про необхідність зробити це обмірковано.
4. фільтри для деревовидної таблиці з інформацією про відвідування URL з хоста за заголовком та URL.

   Вкладка "Пошукові запити" має містити таблицю, яка:

- містить поля: запит, статус, дата, кількість запитів, пошуковик;
- відображає всі дані без сторінок;
- дозволяє реалізувати текстовий пошук по запитам;
- дозволяє робити вибірку по окремому статусу та пошуковику;
- дозволяє проводити багатопріоритетне сортування.

Вкладка "Про користувача" має містити 4 таблиці:

- параметри акаунта Edge;
- збережені паролі (візуалізувати не потрібно, але при необхідності вони можуть бути скопійовані в буфер обміну);
- мовні вподобання (мова, кількість переглядів, кількість перекладів),
- закладки.

Вкладка "Профіль" має містити таблицю "Зміст досліджуваного профілю" з інформацією про файли профілю, з яких отримано інформацію з такими полями: назва елементу, шлях, тип (папка/JSON/SQLite), дата створення та останньої зміни, хеш файла в форматі SHA-256.

При проєктуванні графічного інтерфейсу будемо орієнтуватись на застосування на десктоп моніторах. Це означає наявність монітору, який працює в альбомній розгортці з відношенням роздільної здатності близьким до 16:9.

Як базу візьмемо класичний інтерфейс з білим фоном. Шрифт – Roboto або будь-який інший sans-serif. Базовий розмір– 16 px (~12 pt в нормальному масштабі). Стандартний колір шрифта – чорний. Стандартний колір елементів впливу – світлоголубий (#1CA1C1). Стандартний колір рядку заголовків таблиці – світлофіолетовий (#F4F5F9).

У цілому за базу візьмемо стандартне оформлення компонентів з таблиці стилів webix без особливих налаштувань. Як джерело графічних іконок будемо застосовувати

іконки з колекції Material Design (mdi) [11]. Як джерело іконок, реалізованих за допомогою шрифта, будемо застосовувати іконки з колекції Font Awesome (fa) [12].

Ми маємо розробити 4 екрани, які будуть в межах одного вікна. Для перемикання між екранами використаємо вкладки (Tabs), проте не стиснені, а розгорнуті на весь екран. Ці вкладки будуть розміщуватись зверху 4 екранів, підкреслення буде ідентифікувати саме на якій вкладці ми зараз знаходимось.

Перевірка роботи застосунку на тестовому профілі. На комп'ютері з початку травня 2025 р. була встановлена операційна система Windows з браузером Edge, яка надалі оновлювалась автоматично. Профіль накопичувався з травня до кінця вересня 2025 р. В той час працювала 132 версія браузера. Протягом зазначеного періоду переглядались новини, читалась пошта, проводилась покупка техніки, робились пошукові запити та відвідувались сайти з метою опановування асемблера x86, системи верстки Typst. З кінця серпня частково в цьому браузері шукалась та читалась інформація, пов'язана за тематикою магістерської роботи.

Після збору інформації були сформовані два JSON файли, які містять інформацію про 216 хостів та 103 пошукових запита.

Далі з цими даними можна працювати через візуальну частину застосунку.

Копія екрана з вікном користувача після запуску візуальної частини застосунку показана на рис. 3.

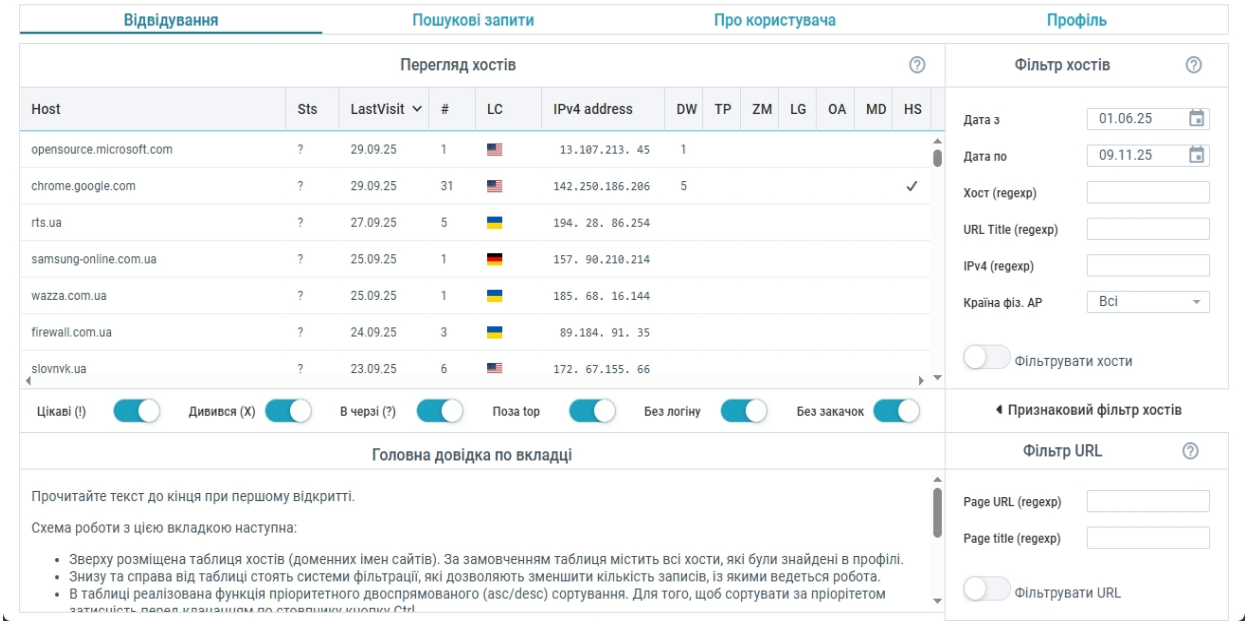


Рис. 3. Вікно інтерфейсу користувача після першого завантаження інформації, зібраної з профіля

В першому рядку показаний загальний заголовок застосунку з його назвою та інформацією про розробника.

В другому рядку показані доступні вкладки (таби) й відмічено що зараз відкрита вкладка "Відвідування".

В третьому рядку розміщена сітка 5x2. В подальшому будемо посилатись на її елементи як (рядок, стовпець).

В (1,1) і (1,2) сітки розміщені заголовки фреймів. В (2,1) розміщена таблиця з повним переліком хостів.

В заголовку таблиці "LastVisit" показано спрямування вниз, що означає сортування в зворотному (desc) порядку. IP адреси відображені так, щоб було зручно виділяти октет чи групи октетів в адресі (сортування за потребою проводиться послідовно за октетами). Бачимо, що з двох хостів були скачані файли (DW), а при

введенні chrome.google.com застосовувалась підказка в адресному рядку, тобто адреса вводилась вручну.

В (2,2) розміщені небінарні фільтри таблиці хостів, за замовчуванням вони всі відключені одним вимикачем. Інтервал встановлюється за замовчуванням від першого знайденого запису до поточної дати.

В (3,1) розміщені бінарні фільтри таблиці хостів. Вони реалізовані як окремі вимикачі й за замовчуванням всі знаходяться в стані "увімкнено" (1).

В (4,1) розміщено інформаційне повідомлення для користувача, приведене в повному вигляді в попередньому підрозділі.

В (4,2) розміщені фільтри для URL. Вони також, як і в (3,1) мають один вимикач.

Для відмічення цікавих сайтів середи тих, в яких була авторизація, встановимо фільтр для їх відображення. Відмітимо, наприклад, не знімаючи фільтр, як цікаві хости discord.com і rozetka.com.ua, а як нецікаві – my.plag.com.ua і account.ukr.net. В результаті ці зміни будуть зафіксовані на екрані й відновляться при наступному запуску. Результат вибірки показаний на рис. 4.

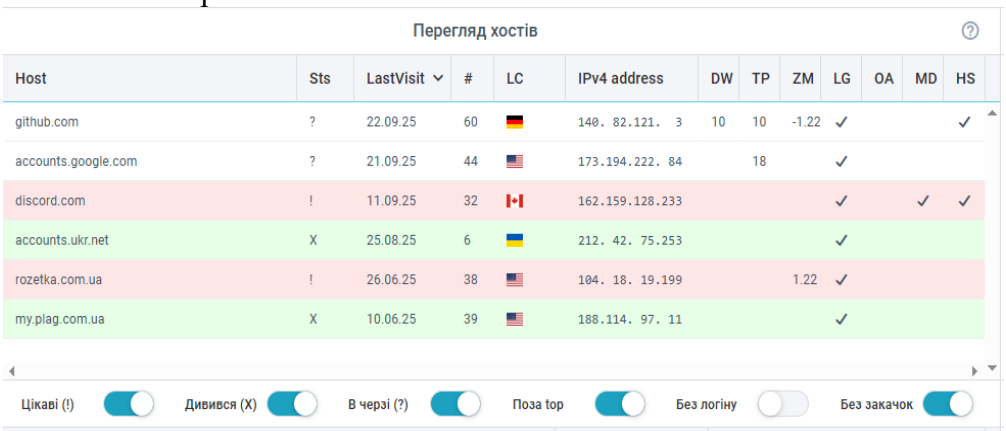


Рис. 4. Вибірка хостів, в яких була авторизація та встановлення оцінок цікавості інформації

Виберемо за допомогою фільтру хостів всіх хости, які відповідають регулярному виразу ^n.*?soft. Результат вибірки показаний на рис. 5

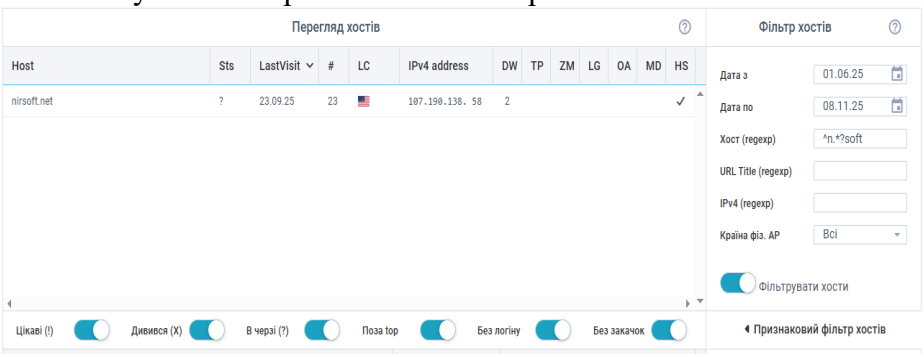


Рис. 5. Відображення всіх хостів, які відповідають регулярному виразу ^n.*?soft.

Натиснемо на ім'я хоста й переглянемо нижній фрейм. Результат показаний на рис. 6.

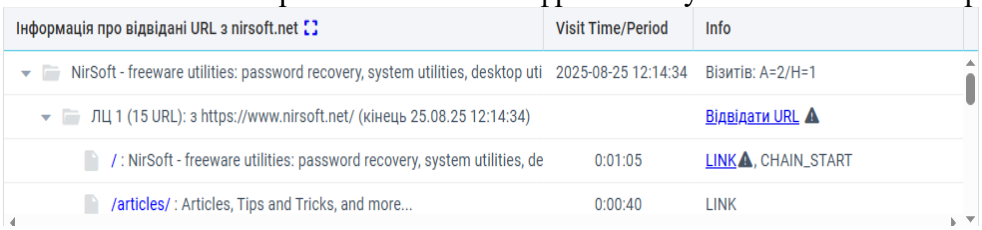


Рис. 6. Відображення інформації про відвідування URL в нижньому фреймі вкладки

Перемкнемось в повноекранний режим, натиснувши на іконку в заголовку таблиці й відкриємо дерев першого ланцюжка, результат за яким показаний на рис. 7.

Інформація про відвідані URL з nirsoft.net	Visit Time/Period	Info
ЛЦ 1 (15 URL): з https://www.nirsoft.net/ (кінець 25.08.25 12:14:34)		Відвідати URL
/ : NirSoft - freeware utilities: password recovery, system utilities, desktop utilities	0:01:05	LINK, CHAIN_START
/articles/ : Articles, Tips and Tricks, and more...	0:00:40	LINK
/articles/view-edge-history.html : View Edge Web browser history with BrowsingHistoryView tool	0:01:10	LINK
/utils/browsing_history_view.html : BrowsingHistoryView - View browsing history of your Web browsers	0:01:50	LINK
/utils/chrome_cache_view.html : BrowsingHistoryView - View browsing history of your Web browsers	0:01:32	LINK
/utils/chromecacheview.zip : успішно 542 кб	0:00:04	Скопіювати в буфер обміну
/web_browser_tools.html : Web Browser Tools Package	0:02:15	LINK
/utils/chrome_history_view.html : ChromeHistoryView - View the browsing history of Chrome Web brow	0:01:11	LINK
/utils/chromehistoryview.zip : успішно 692 кб	0:00:05	Скопіювати в буфер обміну
/utils/chrome_history_view.html : ChromeHistoryView - View the browsing history of Chrome Web brow	0:00:05	BACK
/web_browser_tools.html : Web Browser Tools Package	0:0:19	BACK
/utils/my_last_search.html : My Last Search Package	0:00:30	LINK
/web_browser_tools.html : Web Browser Tools Package	0:00:19	BACK
/computer_forensic_software.html : Computer forenistic software	0:02:10	LINK, CHAIN_END

Рис. 7. Перегляд ланцюжка відвідування URL хоста nirsoft

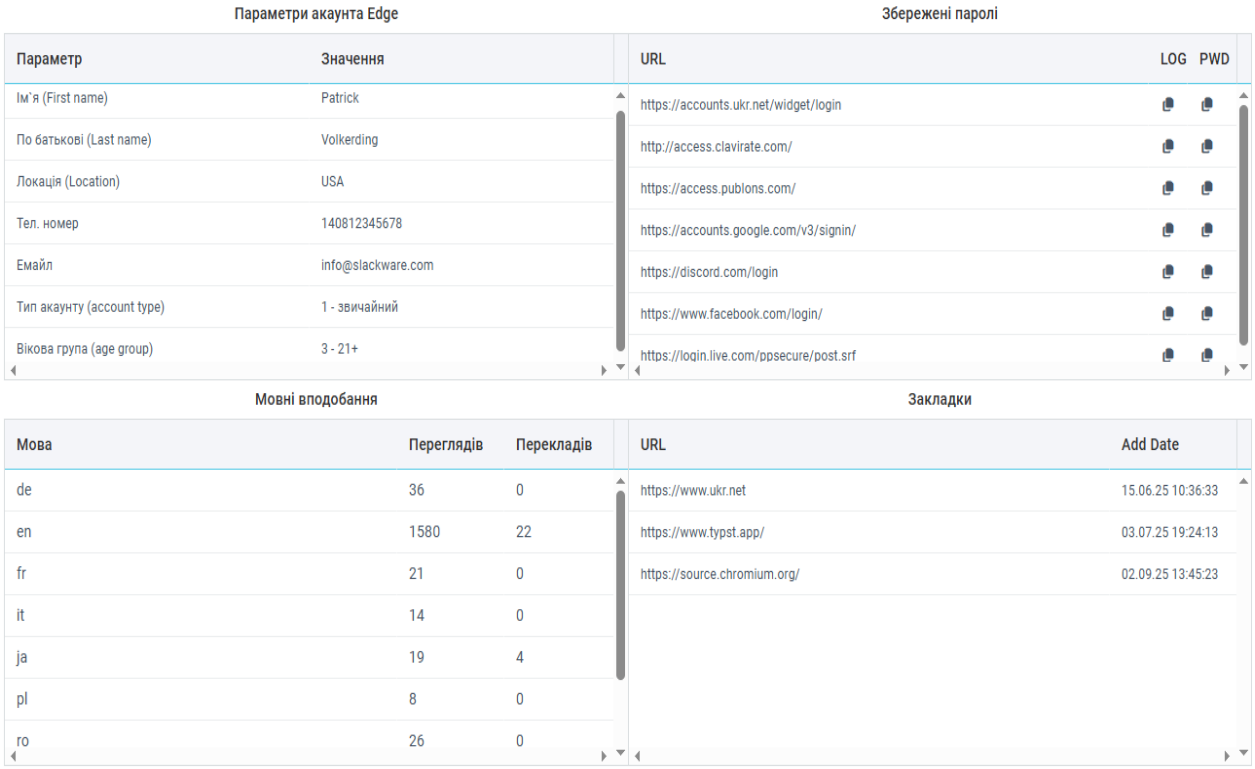
Перемкнемо поточну вкладку на "Пошукові запити", встановимо фільтр "typst" й відмітимо всі записи як нецікаві. Результат запиту показаний на рис. 8.

Перегляд пошукових запитів

Запит	Sts	Дата	#	Пошуковик
typst				
bib to dict typst	X	01.09.25	1	google.com
join sequence typst	X	03.09.25	1	google.com
page numbers typst	X	04.09.25	1	google.com
typst 1.5 line spacing	X	31.08.25	1	google.com
typst app	X	06.09.25	1	google.com
typst bibliography post edit	X	01.09.25	1	google.com
typst check block paragraph	X	26.08.25	1	google.com
typst create package	X	06.09.25	1	google.com
typst dash	X	04.09.25	1	google.com
typst equation align left	X	04.09.25	1	google.com
typst link properties	X	26.08.25	1	google.com

Рис. 8. Результат позначення пошукових запитів користувача, які містять слово "typst" як нецікаві

Зміст вкладки "Про користувача" показаний на рис. 9. Система зібрало достатньо інформації – не тільки телефон й емейл (які мали бути на час реєстрації дійсними, замазані для безпеки), але й розшифрувала ряд паролів (їх можна скопіювати в буфер).



Параметри акаунта Edge

Параметр	Значення
Ім`я (First name)	Patrick
По батькові (Last name)	Volkerding
Локація (Location)	USA
Тел. номер	140812345678
Емайл	info@slackware.com
Тип акаунту (account type)	1 - звичайний
Вікова група (age group)	3 - 21+

Збережені паролі

URL	LOG	PWD
https://accounts.ukr.net/widget/login		
http://access.clavirate.com/		
https://access.publons.com/		
https://accounts.google.com/v3/signin/		
https://discord.com/login		
https://www.facebook.com/login/		
https://login.live.com/ppsecure/post.srf		

Мовні вподобання

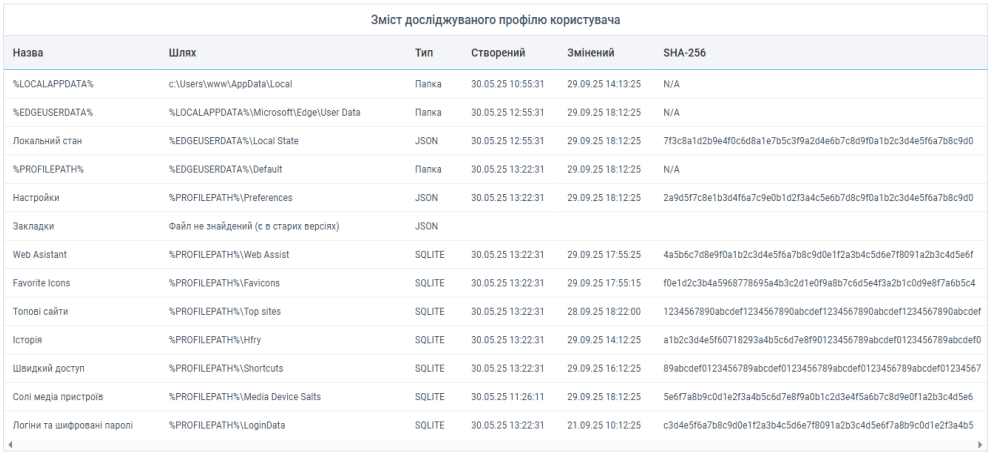
Мова	Переглядів	Перекладів
de	36	0
en	1580	22
fr	21	0
it	14	0
ja	19	4
pl	8	0
ro	26	0

Закладки

URL	Add Date
https://www.ukr.net	15.06.25 10:36:33
https://www.typst.app/	03.07.25 19:24:13
https://source.chromium.org/	02.09.25 13:45:23

Рис. 9. Зміст екрана "Про користувача"

Нарешті, зміст вкладки "Профіль" показаний на рис. 10. Наявність хешів та дат файлів дозволить захиститись від підміни профілю.



Зміст досліджуваного профілю користувача

Назва	Шлях	Тип	Створений	Змінений	SHA-256
%LOCALAPPDATA%	c:\Users\www\AppData\Local	Папка	30.05.25 10:55:31	29.09.25 14:13:25	N/A
%EDGEUSERDATA%	%LOCALAPPDATA%\Microsoft\Edge\User Data	Папка	30.05.25 12:55:31	29.09.25 18:12:25	N/A
Локальний стан	%EDGEUSERDATA%\Local State	JSON	30.05.25 12:55:31	29.09.25 18:12:25	7f3c8a1d2b9e4f0c6d8a1e7b5c3f9a2d4e6b7c8d9f0a1b2c3d4e5f6a7b8c9d0
%PROFILEPATH%	%EDGEUSERDATA%\Default	Папка	30.05.25 13:22:31	29.09.25 18:12:25	N/A
Настройки	%PROFILEPATH%\Preferences	JSON	30.05.25 13:22:31	29.09.25 18:12:25	2a9d5f7c8e1b3d4f6a7c9e0b1d2f3a4c5e6b7d8c9f0a1b2c3d4e5f6a7b8c9d0
Закладки	Файл не знайдений (є в старих версіях)	JSON			
Web Asistant	%PROFILEPATH%\Web Assist	SQLITE	30.05.25 13:22:31	29.09.25 17:55:25	4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8091a2b3c4d5e6f
Favorite Icons	%PROFILEPATH%\Favicons	SQLITE	30.05.25 13:22:31	29.09.25 17:55:15	f0e1d2c3b4a5968778695a4b3c2d1e0f9a8b7c6d5e4f3a2b1c0d9e8f7a6b5c4
Топові сайти	%PROFILEPATH%\Top sites	SQLITE	30.05.25 13:22:31	28.09.25 18:22:00	1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef
Історія	%PROFILEPATH%\Hfry	SQLITE	30.05.25 13:22:31	29.09.25 14:12:25	a1b2c3d4e5f60718293a4b5c6d7e8f90123456789abcdef0123456789abcdef0
Швидкий доступ	%PROFILEPATH%\Shortcuts	SQLITE	30.05.25 13:22:31	29.09.25 16:12:25	89abcdef0123456789abcdef0123456789abcdef0123456789abcdef01234567
Солі медіа пристроїв	%PROFILEPATH%\Media Device Salts	SQLITE	30.05.25 11:26:11	29.09.25 18:12:25	5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6
Логіни та шифровані паролі	%PROFILEPATH%\LoginData	SQLITE	30.05.25 13:22:31	21.09.25 10:12:25	c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8091a2b3c4d5e6f7a8b9c0d1e2f3a4b5

Рис. 10. Зміст вкладки "Профіль"

Таким чином, усі сформовані вимоги застосунок виконує. Тестування показало працездатність усіх функцій, що продемонстровано на рисунках.

Висновки. Проведена розробка застосунку для криміналістичного аналізу профілю браузера Microsoft Edge. В основу застосунку покладена хост-орієнтована модель представлення даних. На відміну від оригінальної, URL-орієнтованої структури моделі даних платформи Chromium, ця модель агрегує всі розрізнені артефакти (історію, файли cookie, дані форм, кеш) навколо імені хоста. Це значно підвищує зручність, швидкість та глибину аналізу, дозволяючи аналітику-криміналісту отримати повну картину взаємодії користувача з кожним сайтом. Таким чином, застосунок розділяється на дві частини: частини збору та агрегування інформації та частину візуалізації з функціями пошуку, яка працює не з даними профілю, а з агрегованою інформацією. Перша частина застосунку написана мовою Python й реалізує набір алгоритмів для вилучення, очищення,

перетворення та агрегації даних з профілю браузера у описаний специфікований формат JSON. Модуль враховує реальні проблеми цілісності даних, усуває неінформативний «шум» (локальні адреси, дані розширення) та додає інформацію, пов'язану з геолокацією хоста. Друга частина реалізує вебінтерфейс для візуалізації та аналізу отриманих даних. Обрана як база для інтерфейсу комплексна бібліотека компонентів Webix підтвердила свою ефективність для роботи з великими обсягами даних (до 50 000 записів) без значних затримок при зміні фільтрів. Проведене тестування підтвердило, що розроблений застосунок повністю відповідає вимогам, сформульованим виходячи з аналізу переваг й недоліків існуючих інструментів. Застосунок призначено для фахівців з кібербезпеки та правоохоронних органів України. Його використання дозволить пришвидшити процес криміналістичного аналізу, забезпечуючи більш повне та наочне представлення цифрових доказів, що важливо для судового процесу. Подальшим розвитком застосунку може бути адаптація до інших браузерів на базі Chromium, окремий аналіз діяльності розширень (зараз вона вилучається), аналіз синхронізованих не декількох пристроях профілів, аналіз змісту закешованих даних браузера, аналіз змісту хосту без його відвідування користувачем з застосуванням ШІ.

Список літератури

1. Відін М.В., Стопакевич О.А., Стопакевич А.О. Аналіз проблеми криміналістичного дослідження історії веббраузера. *Інформатика та математичні методи в моделюванні*. 2025. Т. 15, №. 4. С. 518-534.
2. Chromium design documents. URL: https://www.chromium.org/developers/design-documents/
3. Zacks S. Edge vs. Chrome: Two Great Browsers, but Which is Best in 2025? *Private Internet access*. 2025. URL: https://www.privateinternetaccess.com/blog/edge-vs-chrome/
4. 6 Lesser-Known Chromium Based Browsers Worth Exploring. *MoreLogin*. 2025. URL: https://www.morelogin.com/blog/chromium-based-browser
5. Somé D.F., Airan M., Durumeric Zakir, Staicu C.-A. User Profiles: The Achilles' Heel of Web Browsers. *arXiv*. 2025. DOI: 10.48550/ARXIV.2504.17692
6. Chand R.R., Sharma N.A., Kabir M.A. Advancing Web Browser Forensics: Critical Evaluation of Emerging Tools and Techniques. *Springer Science Computer Science*. 2025. V. 6. P. 355. DOI: 10.1007/s42979-025-03921-6
7. Akintola G.B. Performance Evaluation of four Different Forensic Tools for Web Browser Analysis. *Int. J. of Scientific Research in Multidisciplinary Studies.* 2024. V. 1, No. 10. P.68-82.
8. Adamu H., Ahmad A.A., Hassan A., Gambasha S.B. A Survey of Web Browser Forensic Tools: Autopsy, BHE and NetAnalysis. *Int. J. of Research and Innovation in Applied Science*. 2021. Vol. 6. No. 5. P. 103-107.
9. Chenrnyk M. Webix JS DataTable Best-in-class Performance. URL: https://blog.webix.com/webix-js-datatable-best-in-class-performance/
10. Сергієнко А. В., Балалаєва О. Ю., Гребенькова А. В. Розробка веб-додатку для обліку фінансів та господарської діяльності за допомогою бібліотек Webix UI Library та DHTMLX для системи на основі М-технологій. *Наука та виробництво*. 2023. №25. С. 75-86. DOI: 10.31498/2522-9990252023286712
11. Material Design Icon Collection – Pictogrammers. URL: https://pictogrammers.com/library/mdi/
12. Classic Solid Style icons – Font Awesome. URL: https://fontawesome.com/search?f=classic&s=solid&o=r

DEVELOPMENT OF AN APPLICATION FOR FORENSIC ANALYSIS OF WEB BROWSER HISTORY

M.V. Vidin[1], O.A. Stopakevych[1], A.O. Stopakevych[2]

[1]National Odesa Polytechnic University
1, Shevchenko Ave., Odesa, 65044, Ukraine
[2]State University of Intellectual Technologies and Telecommunications
1, Kuznechna, Odesa, 65023, Ukraine
Email: stopakevich@gmail.com

Web browsers are one of the main sources of digital evidence in modern forensics. Existing tools for forensic analysis of web browser profiles are not effective enough. This is because they reproduce the data model embedded in browsers. This model is focused on rendering speed and optimization of work with individual URLs. Consequently, forensic experts must work with "raw" lists containing thousands of entries, spending considerable time recreating the sequence of user actions and manually filtering technical redirects, trackers, advertisements, and service requests. This study aims to develop algorithms and software applications for the forensic analysis of Microsoft Edge web browser history. These will transform fragmented profile data into structured information suitable for rapid incident detection and user action reconstruction. The development is based on transitioning from the classic, URL-oriented model characteristic of the Chromium database structure to a host-oriented data representation model. This approach allows significant activity artifacts, such as visit history, cookies, and downloaded files, to be aggregated in chronological order around a unique host name. The architectural solution is divided into two parts: a data collection and aggregation module implemented in Python using asynchronous requests for fast processing of large amounts of information, and a visualization module based on the Webix library, which ensures high interface performance when working with large data sets. The application's graphical interface allows experts to perform multi-priority sorting and filtering of records according to multiple criteria. Particular attention is paid to the visualization of transition chains, which makes it possible to recreate the sequence of actions of a suspect on a specific resource. Testing has confirmed the application's ability to quickly process data and identify relevant evidence, significantly reducing the time required for examination compared to conventional manual analysis of “raw” data. It provides a clear representation of digital evidence and minimizes the likelihood of missing important information during investigations.

Keywords: forensics; analysis; browser; history; profile; aggregation; application; host-oriented data model; SQLite; JSON; Python; JavaScript; Webix.