DOI 10.15276/imms.v15.no4.518
УДК 004.056.53

Informatics and Mathematical Methods in Simulation
Vol.15 (2025), No. 4, pp. 518-535

АНАЛІЗ ПРОБЛЕМИ КРИМІНАЛІСТИЧНОГО ДОСЛІДЖЕННЯ 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

Проаналізовано можливості наявних програмних засобів для криміналістичного аналізу історії найбільш застосованих браузерів, побудованих на базі платформи Chromium (Chrome, Edge, Opera тощо). Формалізована та проаналізована наявна схема збереження даних про користувача та його дій в профілі Edge, виявлені місця збереження різних настройок й файлів цього браузера. На основі аналізу вихідного коду Chromium та фактичних збережених даних браузером Edge надані пояснення та зв’язки між сутностями баз даних SQLite профілю й зміст його основного JSON файлу. Розглянуто наявне програмне забезпечення, яке дозволяє проводити криміналістичний аналіз даних з профілю. Це утиліти для роботи з даними та дампами, спеціалізовані утиліти для збору даних з профілю браузера з позиції одного типу інформації – кеш, історія відвідувань, кукі й більш комплексні засоби аналізу профілів браузерів. Останні дозволяють зібрати найбільшу кількість інформації. Проте у цілому комплексне програмне забезпечення має великий потенціал для покращення, оскільки працювати з наявним не дуже зручно для криміналіста.

Ключові слова: Криміналістичний аналіз, браузер, утілити збору даних, комплексний аналіз активності, шкідливе програмне забезпечення, хронологія злочинної діяльності.

Вступ. Для криміналістичного аналізу треба виконати збір доказів, для чого потрібно аналізувати збережені в браузері дані: відвідані вебсайти, історію пошуку, історію завантажень, файли cookie (кукі) та дані форм [1]. Збір даних ведеться в 4 етапи.

1. Аналіз активності. Дозволяє встановити певні особливості характеру користувача, його/її звички, основну діяльність, інтереси, дозвілля та допомогти виявити певні зачіпки, встановити мотиви, проаналізувати потенційну можливість фізичної наявності користувача у пристрою в певні проміжки часу.

2. Аналіз шкідливого програмного забезпечення. Дані з профілю можуть містить індикатори компрометації, починаючи з підозрілих встановлених розширень й закінчуючи періодичним доступом до підозрілих URL. Якщо браузер був захоплений хакером чи шкідливим програмним забезпеченням, це може бути доказом невинуватості користувача браузера в проведенні незаконних дій.

3. Виявлення хронології злочинної діяльності. Дані з профілю дозволяють відтворити послідовність подій, оцінити масштаби інциденту та його наслідки.

4. Отримання доступу до персональних даних. Паролі, дані для автозаповнення, грошові кишені можуть стати джерелом для збору доказів на різних платформах й можуть доказати виявити співучасників криміналістичної діяльності.

Аналіз має допомогти знайти докази кіберзлочинності, крадіжки інтелектуальної власності, інсайдерських загроз та випадків несанкціонованого доступу, а також може знадобитись для маркетологів, кіберпсихологів, соціологів, розробників систем захисту від ботів [2].

Загальна структура та формат збереження даних в профілі браузера. В профілі зберігається наступна інформація.

1. Відвідувані сайти й результат взаємодії з ними: перелік URL та дати, кукі, збережений кеш (для уникнення повторного завантаження зображень, скриптів тощо, іконки сайтів/favicons), поточна сесія та збережені сесії, копії екранів сайтів для попереднього перегляду, структурована інформація для вебзастосунків (бази даних, перелік настройок, автозбережені файли тощо), цифрові сертифікати.

2. Введена користувачем інформація: пошукові запити й обрані пропозиції, введені в форми дані (ПІБ, телефон, емейл тощо), паролі. Багато браузерів зберігають інформацію про картки, персональні рахунки, але тут стандарту немає.

3. Збережена користувачем інформація: закріплені на вкладках сайти, перелік обраного (bookmarks, favorites), результати завантаження файлів та шлях до них в ФС.

4. Встановлені розширення: перелік, версія, увімкнено чи ні, права доступу, настройки конкретних розширень, словники та додані в них користувачем слова.

5. Метрики та інша статистична інформація: використовується для зворотного зв'язку з розробниками браузера.

6. Настройки браузера, які можуть бути змінені користувачем через графічний інтерфейс або ручним чином (за ім’ям параметрів) в спеціальних переліках, інформація про програмне оточення (операційна система, драйвери, тощо)

Більшість інформації зберігається в формі таблиць бази даних SQLite, кожна БД зберігається в окремому файлі без розширення БД. Менша частина як JSON файли, деякі файли зберігаються в бінарному форматі (IndexedBD). Крім того, профіль містить кеш, багато службової інформації до розширень Microsoft та встановлених розширень. За замовчуванням історія зберігається за три останні місяця, а кеш зберігається до трьох місяців (є функція перегляду закешованої версії сайту шляхом додавання cache: перед URL), проте конкретний період збереження певного об'єкта є питанням складним, оскільки Chromium лише приймає до уваги HTTP заголовки, де вказується чи кешувати контент й наскільки довго, проте зберігає об'єкти виходячи з власних критеріїв. Проте, важливо відмітити, що після 3-х місяців видаляється докладна інформація (власне видаляються записи з БД History), але перелік відвідуваних вебсайтів та період їх застосування з моменту створення профілю можливо відтворити за іншими БД, JSON файлами тощо, котрих в профілі значна кількість.

Матеріал про схеми – це результат аналізу вихідного коду та коментарів до нього в репозиторії [3]. Частково застосовувалась документація для розробників браузерів, проте вона зазвичай містить лише загальне призначення й то не завжди. Тобто в інтернеті відсутня офіційна документація щодо структури та призначення полів баз даних, JSON файлів тощо. Відсутність офіційної документації – це позиція розробників. Вони не гарантують стабільність форматів й можуть змінювати схеми зберігання та алгоритми обробки даних без попередження та узгодження.

Розглянемо схеми баз даних SQLite та JSON файлів на прикладі браузера Microsoft Edge версії 140.0.3485.8 в ОС Windows 10 22H2.

Центральною є БД History, схема якої показана на рис. 1.

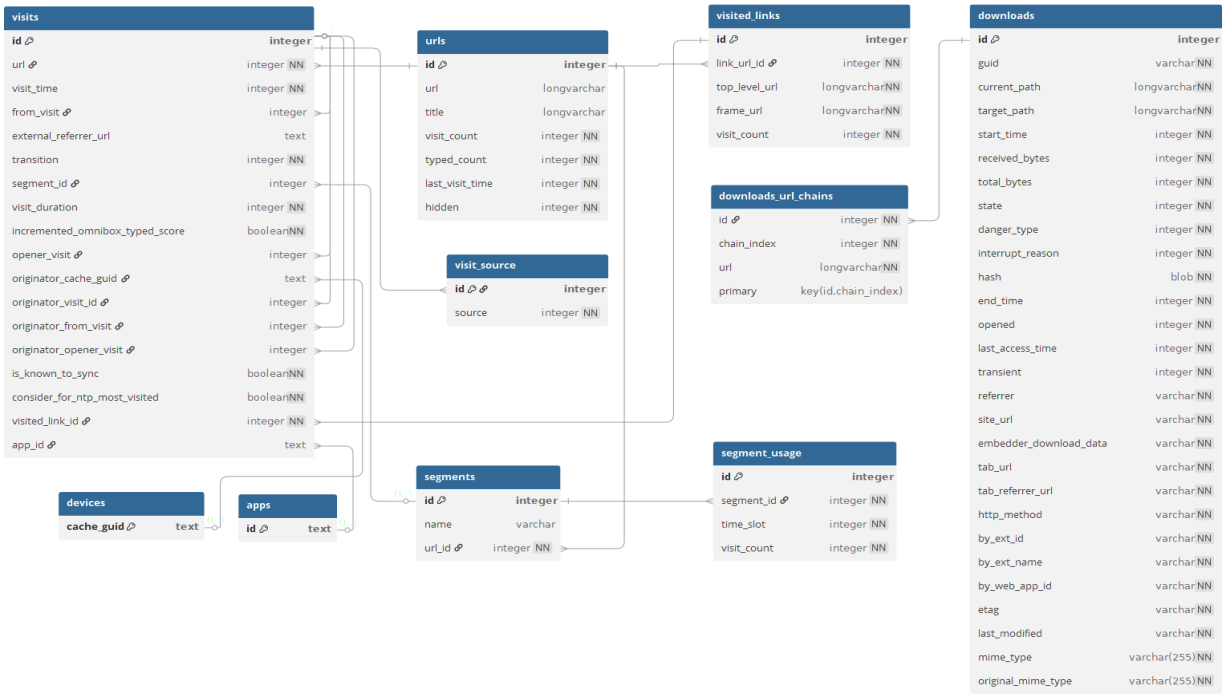

Рис. 1. Схема БД History в Edge (показані тільки важливі таблиці)

Дамо коротку характеристику кожній таблиці: urls – перелік всіх унікальних вебадрес, які відвідувались за час зберігання та кількісна статистика; segments, segments_id – зберігають url в межах внутрішньої класифікації ресурсів браузером (якщо така є); visit_source окрема таблиця для класифікації джерел візитів (див. нижче); visits – власне таблиця, яка зберігає історію усіх відвідувань; visited_links – службова таблиця в якій співставленні URL та ім'я хоста, яке його відображує (наявне в Edge), наприклад, top_level_url = https://86box.net/2025/08/24/86box-v5-0.html, frame_url=https://86box.net/; downloads, downloads_url_chains – зберігає доволі докладні інформацію про збережені файли. Оскільки завантаження – це джерело потрапляння майже всіх файлів на комп'ютер в сучасних умовах, то перевірка файлів проводиться за доволі складною процедурою.

Можливі значення поля кодифікаторів бітового поля visits.transition та їх сенс приведені в табл. 1.

Таблиця 1.

Можливі значення кодифікаторів поля visits.transition (interrupt.chromium\components\ download\ public\common\download_interrupt_reason_values.h)

Код	Назва	Значення
0	LINK	Перехід за покликанням з іншої сторінки
1	TYPED	Введено URL вручну з адресного рядка
2	AUTO_BOOKMARK	Перехід за закладками чи пропозицією з UI
3	AUTO_SUBFRAME	Автозавантаження у фреймі (реклама, віджет)
4	MANUAL_SUBFRAME	Користувач клацнув на щось у фреймі
5	GENERATED	Перехід за згенерованою адресою (наприклад пошук з адресного рядку)
6	AUTO_TOPLEVEL	Автозавантаження стартової сторінки чи сторінки з адресного рядка
7	FORM_SUBMIT	Перехід після відправлення форми
8	RELOAD	Перезавантаження сторінки
9	KEYWORD	Перехід за ключовим словом (користувацька пошукова система)
10	KEYWORD_ GENERATED	Перехід за результатом, згенерованим за пошуком за ключовим словом

Можливі значення поля кваліфікаторів бітового поля visits.transition приведені в табл. 2.

Таблиця 2.

Можливі значення кваліфікаторів поля visits.transition (ui. PageTransition chromium\ui\base\page_transition_types.h)

Код	Назва	Значення
0x01000000	PAGE_TRANSITION_ FORWARD_BACK	Перехід по кнопкам «вперед/назад»
0x02000000	PAGE_TRANSITION_ FROM_ADDRESS_BAR	Перехід з адресного рядка
0x04000000	PAGE_TRANSITION_ HOME_PAGE	Перехід на домашню сторінку
0x08000000	PAGE_TRANSITION_ FROM_API	Перехід ініційований API (наприклад, location.replace)
0x10000000	PAGE_TRANSITION_ CHAIN_START	Початок ланцюжка переходів
0x20000000	PAGE_TRANSITION_ CHAIN_END	Кінець ланцюжка переходів
0x40000000	PAGE_TRANSITION_ CLIENT_REDIRECT	Перенаправлення на стороні клієнта (JavaScript, meta refresh)
0x80000000	PAGE_TRANSITION_ SERVER_REDIRECT	Перенаправлення на стороні сервера (HTTP 3xx)
0x00800000	PAGE_TRANSITION_ IS_REDIRECT_MASK	Маска для перевірки чи є сторінка редіректом

Бачимо, що дані з табл. 1 і 2 містять інформацію, яка дозволяє відтворити історію дуже докладно, аж до натиснення функціональних клавіш в браузері.

БД Top sites. Ця база даних містить одну важливу таблицю top_sites – внутрішній рейтинг браузера вебсайтів за відвідуваністю користувачем, яка має три поля: url, url_rank, title. Кожна адреса має рейтинг від 0 (найчастіше відвідуваний) до орієнтовно 15-20. Поле title – заголовок сторінки.

БД Shortcuts. Використовується для зв'язування пошукових запитів й фактичних відвідувань користувача. Як пошукова система в Chromium браузерах може застосовуватись Google, Bing, DuckDuckGo та ін. Виведений перелік пропозицій, які надає браузер при введені в адресний рядок пошукового запиту називається omnibox. БД містить лише одну важливу таблицю, яка називається omni_box_shortcuts. Коли користувач набирає запит в omnibox, двигун автодоповнення спочатку шукає співпадіння в omni_box_shortcuts. Якщо ярлик не знайдений, то пошук проводиться по таблицям БД History. Схема таблиці omni_box_shortcuts БД Shortcuts має поля: id**,** text, fill_info_edit, url document_type contents contents_class description description_class, type, transition, type, keyword, last_access_time, number_of_hits.

БД Network Action Predictor призначена для прогнозування адреси та для визначення файлів, які будуть завантажуватись з інших ресурсів. Схема БД показана на рис. 2. Таблиця lcp_critical_path_predictor в полі key містить перелік хостів на які користувач заходив переважно сам. Таблиця lcp_critical_path_predictor_initiator_origin містить перелік хостів, з яких переходив часто користувач на інші сайти. Таблиця network_action_predictor містить введений користувачем в адресне поле текст (починаючи з літери) й ті адреси, які були підказані. За кожною адресою фіксується кількість переходів користувача (hits) й кількість разів, коли підказка була не прийнята (misses).

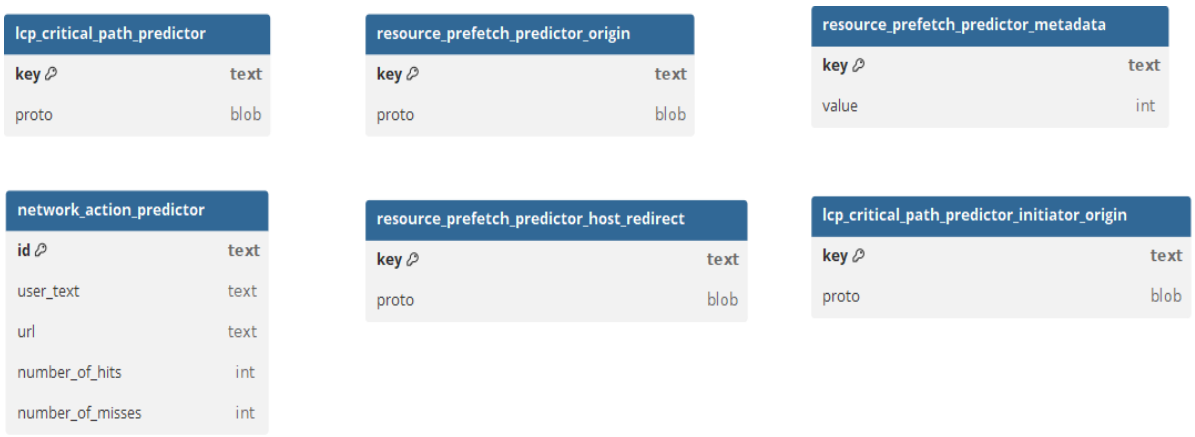

Рис. 2. Схема таблиць БД Network Action Predictor

БД WebAssist (тільки в Edge) має допомагати реалізувати роботу в інтернеті за допомогою голосового управління під час виконання інших дій. Фактично має перелік усіх URL з мета-тегами (не завжди вдало). Зберігає інформацію з початку встановлення браузера й не видаляє її. Схема БД показана на рис. 3.

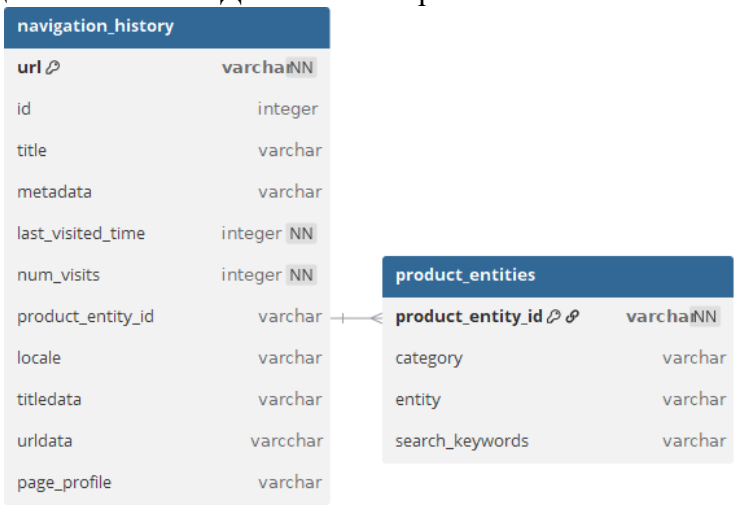

Рис. 3. Схема БД WebAssist

В БД WebAssist наявні дві таблиці, метою яких є класифікація усієї зібраної інформації щодо відвідувань та покупок і інтернеті. Метатеги та інші допоміжна інформація генерується автоматично, приймається до уваги мова ресурсу. Таблиця product_entities містить записи про знайдені товари в інтернеті. Поле entry містить інформацію в вигляді JSON, але стандартної схеми наче немає. Таблиця navigation_history містить перелік всіх відвіданих URL. Вона може використовуватись для створення певного топу за весь період часу, проте ми помітили, що певні сайти в топі не могли стільки відвідуватись, мабуть в якийсь період часу лічильник невірно працював, а може й зараз з ним є проблеми.

Мета БД DIPS дозволяє приймати рішення щодо приватності. Містить одну цінну таблицю bounces. В ній схоже записаний перелік хостів, на яких розміщені власне сайти. Власне оцінка браузера міститься в бінарних файлах DIPS-shm і DIPS-wal. Поле site містить ім’я хоста (www завжди відсутнє), поля first_site_storage_time і last_site_storage_time містять час першого й останнього збереження даних в браузері (включаючи кукі), first_user_activation_time і last_user_activation_time – час першого і останнього використання сайтів. Наявність інтервалу відвідування сайтів – це цінна інформація. Є ще поля, але вони не мають цінності для нашої задачі. Таблиця resource_prefetch_predictor_host_redirect фіксує куди вебсайт звичайно ініціює перехід.

БД favicons призначена для кешування іконок вебсайтів. Відмітимо, що дані не очищуються, тому ця БД теж може стати джерелом інформації про відвідування користувача з часу інсталяції браузера. Звісно, не всі сайти мають іконки, проте значна їх кількість. Цю БД можна розглядати як резервне джерело інформації. Структурна схема БД показана на рис. 4

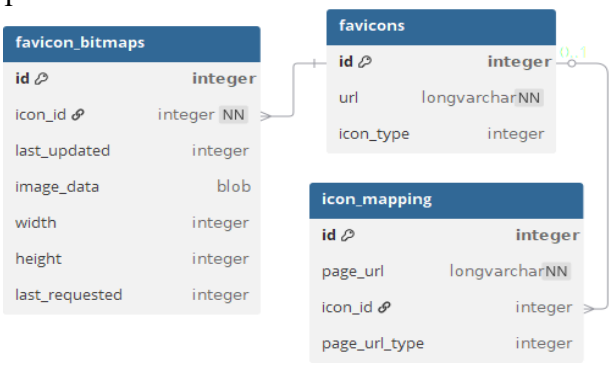

Рис. 4. Схема таблиць БД favicons

Власне іконки зберігаються в favicons_bitmaps.image_data. width і height звичайно або 16x16 або 32x32. Поле icon_type має значення відповідно до ENUM IconType
(0–помилка, 1 – звичайний favicon тощо). Поле icon_mapping.page_url_type відповідає ENUM IconMappingType й визначає до чого відноситься іконка (0 – для однієї сторінки, 1 – до домену чи хосту, 2 – до внутрішніх службових сторінок браузера, 3 – до розширень тощо). Визначення надані в favicon_base chromium\components\ favicon_base\favicon_types.h.

БД MediaDeviceSalts містить солі (в сенсі криптографії) для деяких сайтів, які реалізують голосове введення. Містить одну значущу таблицю media_device_salts. В ній storage_key – URL, salt – соль, creation_time – час створення.

БД Web Data містить понад 50 таблиць, проте майже всі вони пусті або заповнені незначною інформацією. Аналіз переліку та структури таблиць БД, показує, що її метою є збір інформації для автозаповнення. Проте, начебто робиться це не дуже вдало, можливо причиною цього є те, що система орієнтована на англійську мову. Цінна інформація була локалізована лише в 4 таблицях. Таблиця keyword містить перелік пошукових систем загального призначення та на деяких сайтах (на кшталт Wikipedia). Інші таблиці містять введені користувачем й збережені в поля дані. Проте цінність мають далеко не всі записи. В інтернеті присутня одна доповідь, в якій вдалось програмно ідентифікувати особистість на основі цих даних [4]. Дійсно, даних для цього достатньо, проте без технологій ШІ обробити це складно.

БД Login Data містить збережені логіни, паролі й певну статистичну інформацію. Структурна схема БД показана на рис. 5.

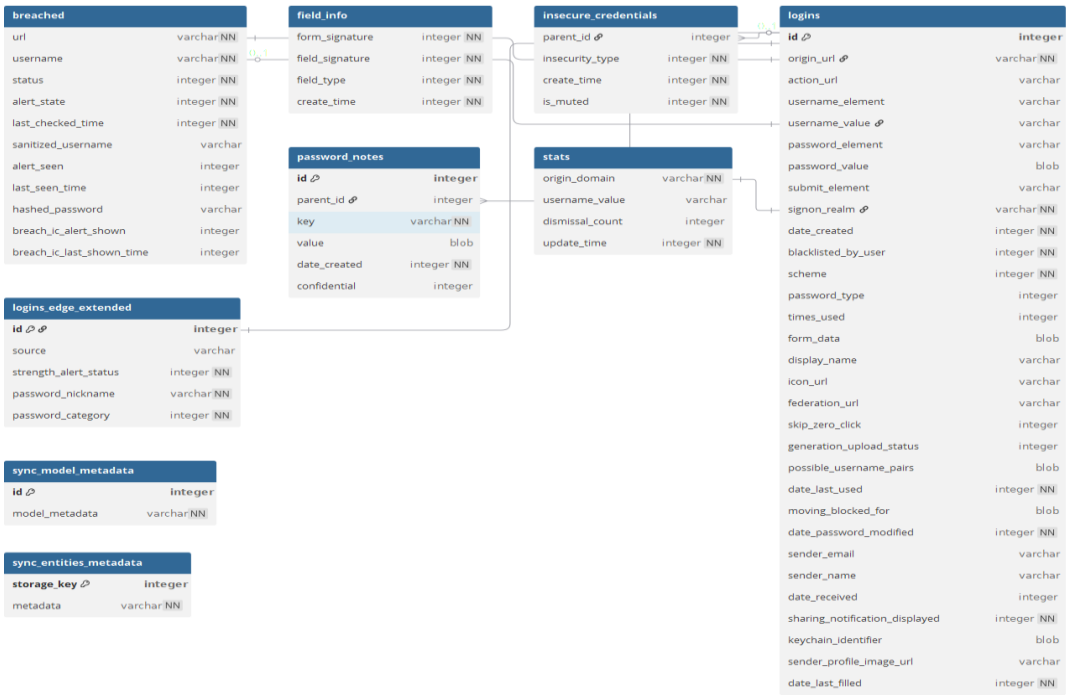

Рис. 5. Схема таблиць БД Login Data

Ключовою таблицею тут є logins. Ця таблиця фактично містить інформацію як авторизуватись в формі авторизації певного сайту. Поле origin_url – це вебадреса сторінки з формою для входу, а action_url – це вебадреса, на яку ця форма вказує. username_element – це ім'я HTML-элемента для логіну й username_value – його значення. password_element – ім'я HTML-елемента для паролю й password_value – це зашифрований пароль (про шифрацию буде описано нижче). submit_element – HTML-ідентифікатор кнопки для відправлення форми. Поле scheme містить тип форми: 0 – звичайна, 1 – проста тощо (див. ENUM Scheme – PasswordForm chromium\components\download\public\common\download_interrupt_reason_values.h).
Поле password_type вказує на тип пароля: 0 – звичайний, 1 – згенерований, імпортований (див. ENUM Type). Поле times_used містить кількість входів. Інші поля не мають великого сенсу, більшість з них завжди пуста. Таблиця logins_edge_extended містить додаткову інформацію, яка використовується лише в Edge. Поле source містить джерело збереження (0 – браузер, 1 – імпорт, синхронізація) Поле strength_alert_status вказує на те, що пароль слабкий. Поле password_category класифікує пароль на 4 категорії:
1 – корпоративний акаунт, 2 – Microsoft Account, 3 – Windows Hello / PIN, застосунок/розширення. Google застосовує спеціальні сервіси для перевірки надійності паролів, паролі які знаходяться в словниках паролів чи відомо, що були вкрадені вказуються як ненадійні. Таблиця breached містить інформацію о знайдених викрадених паролях (інтеграція з Password Monitor / Have I Been Pwned). Поле url – це адреса сайта у якого був вкрадений пароль. Поле username містить ім'я користувача Поле status має три стани: 0 – не перевірено, 1 – перевірено й пароль не в небезпеці, 2 – перевірено й пароль в небезпеці. Поле alert_state містить стан повідомлення про небезпеку (показано/не показано). Поле last_checked_time містить час останньої перевірки. Поле hashed_password містить хеш паролю для звірки. Таблиця insecure_credentials містить відомості про слабкість паролів (слабкі, повторно використані, вкрадені). Поле insecurity_type вказує на тип небезпеки: 0 – похищений (Leaked), 1 – слабкий, 2 – повторне використання, 3 – компрометація через фішинг. Таблиця field_info зберігає метадані про поля форм для автозаповнення, не містить корисної інформації. Таблиця password_notes призначена для зберігання зауважень до паролів. Така функція

передбачена в API Chromium, але не передбачена в графічному інтерфейсі браузера. Наразі таблиця не заповнюється й що має містити замітка – не повністю визначено. Таблиця stats містить статистику відмови від використання автозаповнення. Поле – це домен, username_value – логін, dismissal_count – скільки разів автозаповнення відхилено і update_time – час останнього відхилення. Таблиці sync_entities_metadata і sync_model_metadata містять службову інформацію про синхронізацію паролів між пристроями користувача.

БД Login Data For Account за структурою ідентична Login Data. Активується якщо вмикається синхронізація між пристроями на базі акаунта Microsoft. В протилежному випадку пуста.

БД Visited Links призначена для зберігання хешованих URL. База призначена для прискорення пошуку відвіданих сайтів (наприклад, для зміни властивостей посилань на вже відвідані вебсайти, наприклад кольору). Проте, в ній мають зберігатись не чисті URL, а їх хеші за канонізованими (тобто приведеними до стандартної форми, яка називається в браузері canonical) URL. Вихідний код функцій для хешування та пошуку знаходиться в visitedlink_writer chromium\components\visitedlink\browser\ visitedlink_writer.cc, проте він сильно розбитий по функціям, тому складно відтворити точно логіку. З опису можна зрозуміти, що застосовується метод відкритої адресації або закритого хешування – це метод вирішення колізій у хеш-таблицях. При цьому методі хеш-колізія вирішується шляхом зондування, або перебору альтернативних місць у масиві (послідовність зондування), поки не буде знайдено цільовий запис, або не буде знайдено невикористаний слот масиву, що вказує на відсутність такого ключа в таблиці. Такий метод має обмеження на розмір, тому, можливо, він був замінений з пошуку в хешовій таблиці на диску на альтернативний, який робиться в оперативній пам'яті. Така реалізація теж присутня в програмному коді. Інакше, судячи по коду, якщо є проблеми з файлом хешу й він застарів, браузер має його видалити й створити знову. Але це не робиться. Аналіз файлу показує, що він застосовує 4-байтовий хеш. Більшість з хешів нульові, що пояснюється тим, що застосовується алгоритм лінійного зондування (linear probing). Тобто браузер виділяє місце для зберігання великого масиву по 4 байти й хешує URL, застосовуючи решту від ділення на розмір цього масиву. Наприклад для 256кб записів : hash(url) % 256000. Отримане значення – це індекс, куди треба записати 4-байтовий хеш. Якщо комірка пуста (нульова), браузер записує туди хеш. Якщо ні – шукає наступну вільну комірку. В будь-якому випадку цей файл є вторинним й як виходить з опису будується з БД History. В раціональний час не вдалось реалізувати такий алгоритм так, що працював з базою Chromium – схоже там специфічна реалізація деяких функцій, не така як в бібліотеках Python. Проте сама ідея застосування такого алгоритму доволі продуктивна. Оскільки дані про певний URL знаходяться не тільки в History, а можуть знаходитись (чи не знаходитись) в інших файлах, то застосування такого алгоритму може дійсно підвищити швидкість пошуку при застосуванні циклічної обробки, коли всі хеші будуть завантажені в пам'ять.

Наступні БД BrowsingTopicsSiteData, DashTrackerDatabase, ExtensionActivityEdge, ExtensionActivityComp, Extension Cookies, InterestGroups, PrivateAggregation, ServerCertificate, Shared-Storage не містять цікавої інформації.

JSON файл preferences у профілі браузера Edge є центральним сховищем налаштувань користувача та стану браузера. Він містить як глобальні параметри, так і специфічні дані для окремих сайтів. Основна мета цього файлу – забезпечити відновлення середовища користувача після перезапуску браузера та синхронізацію поведінки між різними пристроями. Структура файлу показана на рис. 6.

У файлі можна виділити кілька ключових груп даних. По-перше, це загальні налаштування інтерфейсу та поведінки (в тому числі експериментальні й не задокументовані): мова інтерфейсу, тема оформлення, стартова сторінка, набір

закріплених вкладок, параметри автозаповнення форм і пошукові системи за замовчуванням. По-друге, у Preferences зберігаються дані про облікові записи: інформація про синхронізацію з акаунтом Microsoft, дозволи на використання паролів, історії та закладок. Тут же фіксуються параметри доступу до хмарних сервісів і службових токенів. Третій блок – специфічні для вебсайтів налаштування. Для кожного домену браузер веде окремі записи, що включають дозволи (доступ до камери, мікрофона, геолокації, сповіщень), а також показники взаємодії користувача із сайтом.

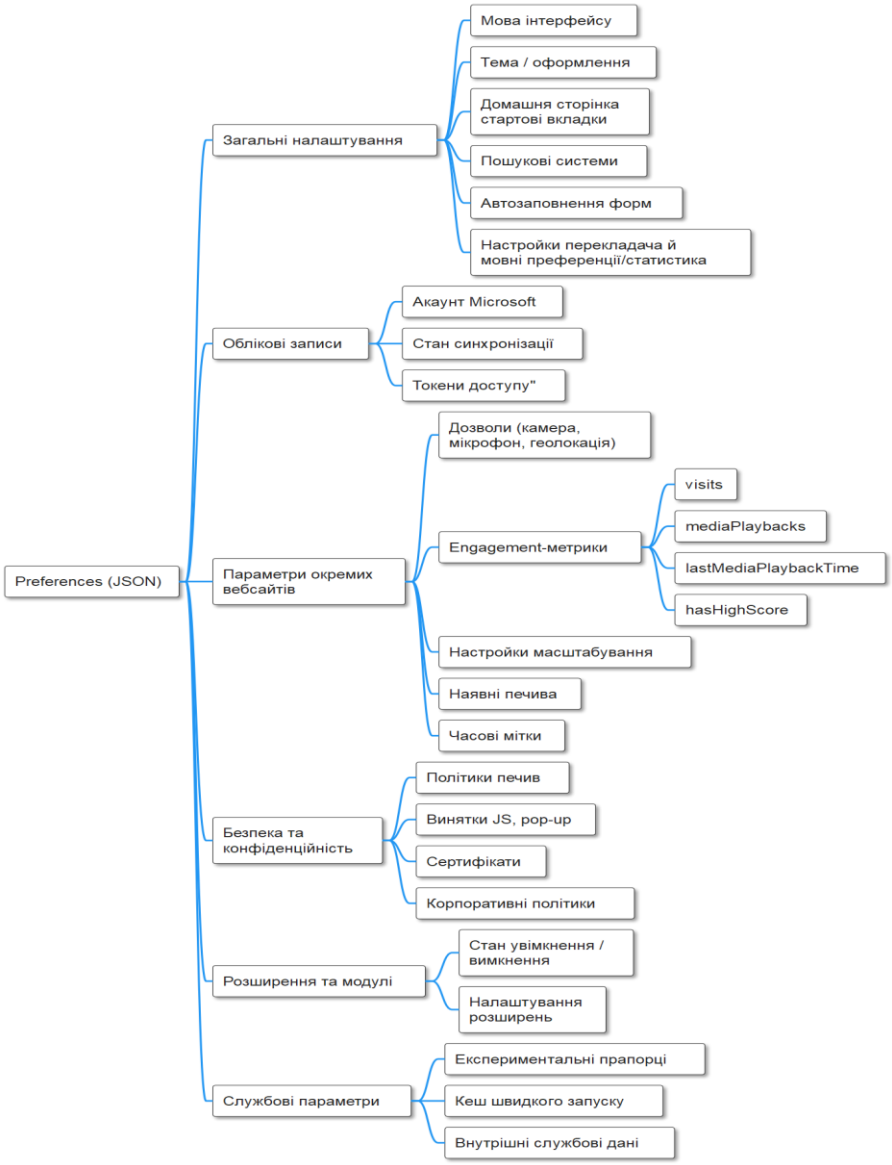

Рис. 6. Схема JSON файлу preferences

Наприклад, кількість відвідувань, факти відтворення медіа, час останньої активності. Ці дані використовуються системою Site Engagement для визначення рівня довіри до ресурсу та пріоритетності його показу. Четвертий блок – системні параметри безпеки та конфіденційності. Тут зберігаються відомості про заблоковані або дозволені куки, політики відстеження, винятки для JavaScript чи спливаючих вікон. Також фіксуються дані про сертифікати, політики корпоративного керування та результати перевірок безпеки.

Для криміналістичного аналізу найбільш цікавий третій блок.

В браузерах на базі платформи Chromium застосовується Site Engagement Service для оцінки рівня зацікавленості користувачів в даних конкретного origin. Origin – це термін Chromium, тобто протокол+субдомен+домен+порт. Субдомен в деяких випадках

браузер змінює на маску [*.] , тобто origin відноситься до всіх субдоменів. Порт відноситься до необов'язкової інформації, в деяких випадках достатній тільки протокол. Кожен вебсайт в цій системі має свій бал, який змінюється в часі за набором певних правил. Бал – число з плавучою точкою від 0 до 100. Активність користувача збільшує бал, але є максимум приросту на день, щоб уникнути «накручування». Якщо сайт не відвідують, бал поступово зменшується з часом. Не синхронізується: оцінки зберігаються локально на пристрої, не передаються між профілями. Крім engagement інформацію щодо origin містять: app_banner – сайти, які використовували API Google для показу банерів; automatic_downloads – сайти, якім дозволено видавати на завантаження більше ніж один файл; client_hints – фіксація сайтів, які запитували спеціальні дані про пристрій та операційну систему користувача. Крім того, можуть представляти певну цікавість мовні вподобання користувача language_usage_count – кількість переглянутих сторінок певною мовою; language_dwell_time_average – кількість часу (в відносних ненормованих одиницях схоже), проведеного на сторінках певною мовою (чомусь використовуються не ISO 639-1, наприклад є статистика по uk й по ua, схоже що браузер спирається на метатеги, які не завжди надають вірну інформацію про мову сторінки); language_model_counters – кількість сайтів по мовам, які визначив браузер за допомогою нейромережевої моделі. В цьому випадку мов істотно менше, ніж в попередньому пункті; language_translated_count – кількість перекладених сторінок з різних мов.

JSON файл Bookmarks містить інформацію про вибрані вебсайти користувача (назва запису, url тощо) й інформація про стан синхронізації.

Наступні JSON файли не містять важливої інформації: JSON файл Secure Preferences містить істотну кількість параметрів, частина з них закодована. Більшість - це розширення від Microsoft (наприклад, біля половини присвячена розширенням для читання голосом різними мовами, при чому для деяких мов є декілька варіантів голосу). Крім того, тут зберігають настройки деякі розширення. JSON файл PrefferedAps схоже, що створений на майбутнє. Зараз не містить нічого крім одного безсенсового запису. JSON файл HubApps містить допоміжну інформацію для чат-бота Copilot. JSON файл BrowsingTopicsState має містить допоміжну інформацію, але нічого крім нульової статистики не містить.

Зупинимось ще на питанні: яким чином користувач може "сховатись" він занесення своєї активності в усі перелічені файли? Всі браузери на платформі Chromium мають режим інкогніто (приватний режим). В цьому режимі відкривається вікно з чистим профілем й після його закриття вся інформація має втрачатись. Основні відмінності між звичайним режимом й режимом інкогніто розглянуті в [5-7]. Історія, паролі, введені дані тощо зберігаються лише в оперативній пам’яті. У цілому, браузер дійсно не буде фіксувати активність користувача поки він працює в цьому режимі й отримати інформацію про його активність з профілю буде не можливо, оскільки фактично користувач працював в іншому, тимчасовому профілі. Багаторічні дослідження фіксують, що рівень захищеності режиму анонімності в Chromium вище, ніж у браузерів попередніх поколінь й зайва інформація не зберігається. Це підтверджується, наприклад, в дослідженні [8], в якому застосовували технологію віртуальних машин для порівняння образів до та після відвідування переліку сайтів в 30 браузерах й більш. Більш свіже репрезентативне дослідження [9] також підтверджує, що рівень анонімності основних браузерів відповідає заявленому.

Проте, звісно, приватний режим сам по собі без VPN та схожих технологій не забезпечує достатній рівень анонімності. Час доступу до сайтів та їх доменне ім’я може зберігатись у провайдера в журналі DNS-серверу чи в журналі інших DNS-серверів. Браузер не змінює свою ідентифікацію: User-agent, часову зону, IP адресу (й внутрішню адреса в NAT). Тому з вебсайту, якщо ведуться належні журнали, можна ідентифікувати користувача.

Застосування загальних інструментів криміналістичного аналізу й аналіз дампів. Місце встановлення основних компонентів браузера за замовчуванням зведено в табл. 3.

Таблиця 3.

Місце встановлення основних компонентів браузера Edge в ОС Windows

Компонент	Шлях
Основна програма	C:\Program Files (x86)\Microsoft\Edge\Application|Містить msedge.exe, бібліотеки, ресурси
Update Engine	C:\Program Files (x86)\Microsoft\EdgeUpdate|Служба оновлення браузера
User Data (профіль)	%LOCALAPPDATA%\Microsoft\Edge\User Data|Тут зберігаються всі профілі, історія, куки, логіни
Кеш	%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Cache|Chromium-кеш, IndexedDB, Media Cache
Розширення	%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Extensions\ Установлені розширення

Попри те, що виконавчий файл встановлюється в каталог для 32-бітних програм, це не обов'язково означає, що msedge буде саме 32-бітним. Звичайно в 64-бітній ОС встановлюється 64-бітний Edge та інші супутні компоненти, проте з використанням корпоративного інсталятора можна зробити різні вибори.

Формально незалежними компонентами від Edge, хоча вони встановлюються й оновлюються разом, є WebView2, widgets та xsocial. WebView2 – це портативна версія Edge, яка призначена для виконання в програмах як двигун графічного інтерфейсу. Інші дві програми – це новини з сайту MSN та для користувачів XBox. Обидві програми вимагають застосування WebView2.

64-бітний Edge застосовує наступні ключі реєстру – глобальні (HKLM), які визначають політики, версію, параметри оновлення, дозволи та користувача (HKCU), які зберігають налаштування користувача, синхронізацію, дозволи, стан профілю.

Програмні інтерфейси Windows для взаємодії з Edge приведені в табл. 4.

Таблиця 4.

Програмні інтерфейсі ОС Windows для взаємодії з Edge

API / Інтерфейс	Призначення
WinINET / WinHTTP	Edge використовує WinHTTP для системних запитів
ShellExecute / COM	Запуск браузера через msedge.exe з параметрами
Group Policy (ADMX)	Керування політиками через GPO
WebView2 Runtime	Вбудовування Edge як движка рендерингу в застосунки
DPAPI	Шифрування паролів у Login Data. Для розшифрування паролів потрібен SID користувача та ключі з Protect в реєстрі
Windows Search / Jump Lists	Інтеграція з пошуком, історією запуску

Реалізовані API оптимізовані під ОС Windows, що робить взаємодію та адміністрування під цією ОС зручнішим ніж, наприклад, з Google Chrome.

Інструменти загального призначення зведемо до табл. 5.

Таблиця 5.

Інструменти загального призначення, придатні для задач криміналістичного аналізу

№	Назва ПЗ	Призначення	Сайт
1	AccessData FTK Imager	Створення образів дисків й пам'яті для їх подальшого криміналістичного аналізу	exterro.com
2	Volatility	Інструмент для аналізу образів дисків і пам'яті, файлу підкачування, гібернації, інших дампів з великою кількістю плагінів для виконання всіх основних задач	github.com/ volatilityfoundation/ volatility
3	DB Browser for SQLite	Зручний графічний інтерфейс доступу до баз даних SqLite з можливістю виконувати довільні команди	sqlitebrowser.org
4	Janice	Програма для перегляду змісту JSON файлів як дерев, оптимізована для роботи з файлами великого розміру	github.com/Janice

продовження табл. 5.

5	ImHex	Багатофункціональна програма для перегляду бінарних файлів (для IndexedDB)	github.com/ WerWolv/ImHex
6	DCode	Програма для декодування різних форм запису дати/часу подій (підтримуються особливі формати зберігання Chromium), може застосовуватись до роботи з бінарними файлами	www.digital-detective.net/dcode/
7	CyberChef	Вебінструмент для декодування, дешифрування, парсингу Застосовується для декодування protobuf, base64, hex, gzip — вони часто зустрічаються в браузерних артефактах	gchq.github.io/ CyberChef
8	ShellBags Explorer	Дозволяє побічно підтвердити доступ до профілю браузера або його копіювання	tzworks.com
9	Registry Explorer	Перегляд реєстру Windows з криміналістичними функціями Може виявити налаштування браузера, політики, сліди запуску ericzimmerman.github.io	

Уявимо, що зроблено дамп активної операційної системи з браузером Edge за допомогою FTK Imager. Як отримати потрібні дані? Можна подивитись в базу даних в пам'яті. Операційна система для ефективності використовує механізм кешування файлів у пам'яті. Це означає, що коли Chrome читає файл History з диска, вміст цього файлу зберігається в оперативній пам'яті (у кеші файлової системи). Під час наступних звернень до файлу система може дуже швидко зчитати дані прямо з ОЗП, не звертаючись до повільного диска. Крім того, сам процес msedge.exe буде зберігати частини цієї БД у своїй власній пам'яті (кучі процесу), оскільки він працює з даними з файлу. В пам'яті буде знаходитись не обов'язково вся БД, але її значна й часто актуальна частина.

Коли створюється дамп пам'яті (наприклад за допомогою FTK Imager, Belkasoft Live RAM Capturer чи DumpIt), виконується наступне. Створюється знімок усієї оперативної пам'яті на даний момент часу, що включає. Дані всіх запущених процесів (виконавчий код, кучі, стеки тощо). Кеш файлової системи, в якому як раз можуть знаходитись блоки (сторінки) даних з файлу History й багатьох інших недавно використаних файлів. Різні інші структури даних ядра ОС.

Розглянемо більш детально застосування Volatility.

Після того, як створено дамп (наприклад в файл memory.raw) необхідно застосовувати спеціальні плагіни в профілі Win10x64.

Для версії 2.x починати треба з наступних плагінів

```
volatility -f memory.raw --profile=Win10x64 pslist
volatility -f memory.raw --profile=Win10x64 dlllist -p
volatility -f memory.raw --profile=Win10x64 cmdline -p
volatility -f memory.raw --profile=Win10x64 netscan
volatility -f memory.raw --profile=Win10x64 chromehistory
```

На що звертати увагу? Процеси msedge.exe — перевірити кількість, PID, час запуску. Відкриті вкладки — через chromehistory, cmdline, dlllist. Мережеві з’єднання — netscan покаже активні TCP/UDP сесії. Завантажені DLL — можуть вказати на розширення або шкідливі модулі. Кешовані URL / історія — витягується з ОЗП, навіть якщо профіль був очищений.

Як працює плагін chromehistory в версії 2.x? Сканує пам'ять - проходить по всьому дампу пам'яті, шукає сигнатури, структури, характерні для процесів Chrome. Знаходить SQLite БД – шукає в пам'яті фрагменти, які мають спеціальних заголовок. З окремих фрагментів в пам'яті намагається відновити цілісну структуру бази даних в пам'яті. Якщо це вдається, то до неї можна робити SQL-запити. Інші плагіни з префіксом “chrome” працюють аналогічно [10]. В версії 3.x концепція пошуку інша. ЇЇ використовує плагін Chrome HX [11]. Плагін НЕ сканує всю пам'ять навмання. Замість цього він використовує потужний механізм Volatility 3 під назвою yarascan. Yarascan

дозволяє шукати в пам'яті за певними шаблонами (YARA-правилами). У цьому разі плагін шукає не просто випадкові дані, а сигнатуру заголовка SQLite бази даних (перші байти файлу History). Коли Volatility знаходить такий заголовок, вона здатна визначити, чи є цей регіон пам'яті відображеним файлом (memory-mapped file). Операційна система, коли програма читає файл, може відобразити його вміст прямо у віртуальну пам'ять процесу. Це ефективний механізм доступу. Плагін витягує цей відображений файл цілком. По суті, він знаходить у пам'яті точну копію файлу History з диска (на момент створення дампа) і зберігає його у вигляді файлу на диск дослідника.

Цікавою є робота [12], в якій запропонована методологія для отримання даних з браузерів на платформі Chromium.

1. Ідентифікація класів і структур, які представляють браузер, вкладку, групу вкладок, відвідану URL-адресу тощо у вихідному коді проекту Chromium.

2. Генерування об'єктної структури таких класів і структур та сканування дампу пам'яті для пошуку об'єктів Browser як відправних точок аналізу пам'яті і, відповідно, виявлення дій користувача на основі Chromium.

3. Нарешті, виявивши об'єкт Browser, можна інтерпретувати значущі поля і дослідити всі доступні підоб'єкти, використовуючи змінні-покажчики, з метою вилучення значущої з точки зору криміналістичної експертизи інформації, наприклад, про відкриті вкладки і відвідані URL-адреси. Спрощено ідея роботи [22] полягає в тому, що програма мовою C++, скомпільована Visual C++ має певну структуру: не тільки типи даних стандартизовані в розмірах, але й структура класів, відношень між класами, формат переносу за слово за допомогою вказівників тощо. Тобто ми можемо відрізнити де йде клас, а де інший код. А значить ми можемо шукати структуру, яка пов'язана власне з класами, як вони реалізовані в конкретній редакції Chromium. Таким чином, необхідно обрати відповідну певному браузеру (Edge, Chrome тощо) версію Chromium й її скомпілювати в режимі збереження символів для відлагодження. Маючи символьні pdb файли ми можемо знайти в коді дампа місце, яке відповідає структурі полів класів. Структуру цих класів розробники браузерів на основі Chromium не змінюють.

Важливі класи та їх поля в коді Chromium приведені в табл. 6.

Таблиця 6.

Важливі класи та їх поля в коді Chromium

Клас	Поле	Коментар
Browser	profile_	Ім'я профілю чи профіль приватний
	tab_strip_model_	Вкладки й групи вкладок
	session_id_	ID об'єкта класу Browser
	bookmark_bar_state_	Чи показувати панель закладок
	window_has_shown_	Чи вікно показується
	user_title_	Заголовок вікна
ProfileImpl	path_	Шлях до AppData
TabStripModel	contents_data_	Перелік створених вкладок
group_model_	Дані про групи вкладок	
selection_model_	Індекс останньої використаної вкладки	
TabGroup	id_	ID групи вкладок
	visual_data_	Параметри відображення групи вкладок
	tab_count_	Кількість вкладок в групі вкладок
TabGroupId	token_	16-байтовий масив, який представляє унікальний ID
TabGroupVisualData	title_	Заголовок групи вкладок
	color_	Колір групи вкладок
NavigationControllerImpl	entries_	Перелік відвіданих URL
NavigationEntryImpl	frame_tree_	Дані, пов'язані з HTTP запитом
	unique_id_	ID запису
	title_	Заголовок відвіданої вебсторінки

продовження табл. 6.

	favicon_	Інформація про іконку сайта
	ssl_	дані SSL (сертифікат тощо)
	user_typed_url_	URL, що показується в адресному рядку
	timestamp_	Час відвідування (Chrome Epoch)
	http_status_code_	Код HTTP відповіді

Структурно ієрархію класів представимо на рис. 7.

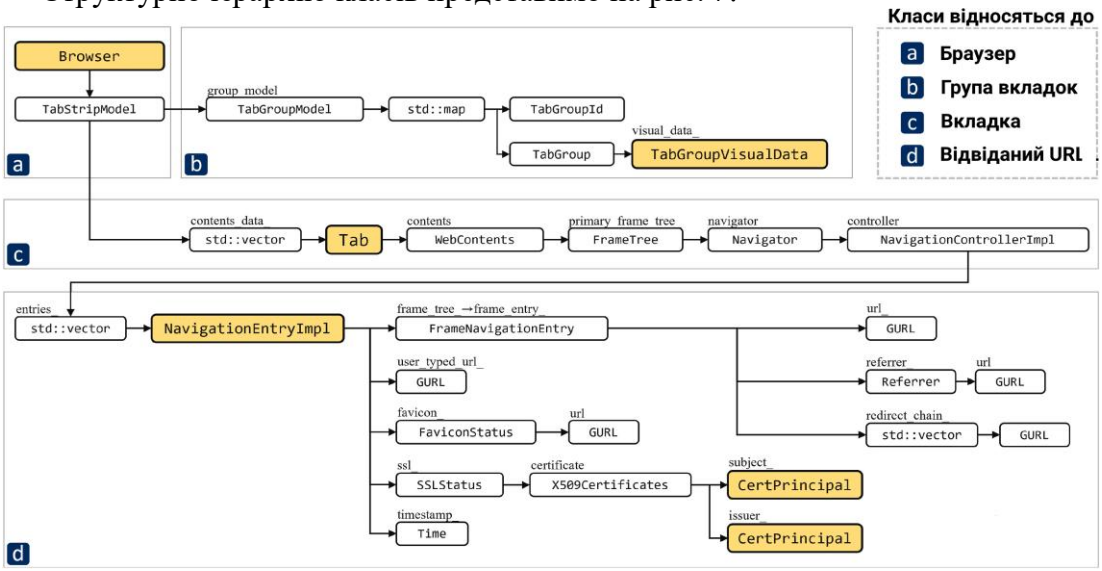

Рис. 7. Ієрархія ключових класів (адаптовано з [22])

Автори показали, що застосування алгоритму дозволяє визначити історію відвідувань за кожною вкладкою вікна. При чому не важливо, це приватний профіль чи звичайний. Для застосування підходу достатньо зробити дамп активного процесу msedge (через диспетчер задач чи програму procdump) без його зупинки й провести пошук за алгоритмом. Файл дампа вікна Edge звичайно не менше 0.5 Гб й більше 2 Гб якщо робити повний за допомогою procdump. Приклад знайденої частини, яка може бути заголовком вкладки в режимі Private показаний на рис. 8.

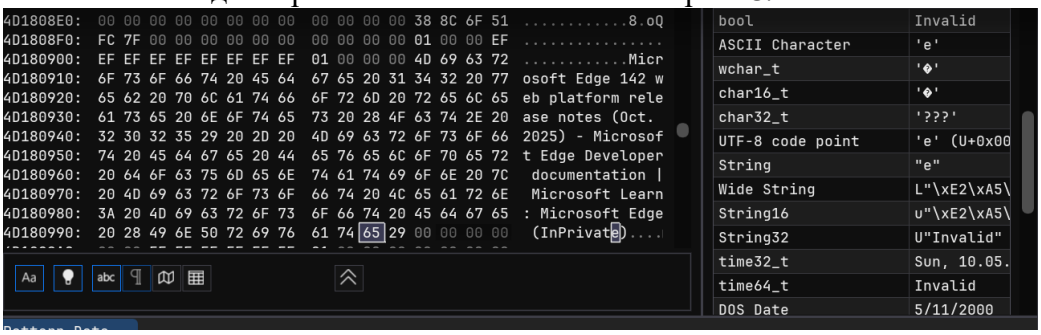

Рис. 8. Пошук в редакторі InPrivate текстового рядку в форматі UTF-8

Таким чином, загальні інструменти потенційно дозволяють більше ніж можна отримати з баз даних профіля. Проте це набагато складніше. Певний сенс вони мають лише коли наявний дамп, але не самі бази профіля або коли потрібно зібрати інформацію з дампу запущеного браузера з наявними вкладками в приватному режимі.

Застосування спеціалізованих засобів аналізу профілів браузерів на платформі Chromium. Далі розглянемо більш спеціалізовані інструменти.

BrowsingHistoryView від NirSoft [13]. Універсальна програма для перегляду історії . Достоїнством застосунку є те, що він може працювати майже з усіма

сучасними й не дуже браузерами під ОС Windows. Можливо зібрати URL, дату відвідування, кількість відвідувань й адресу, з якої потрапили на сторінку.

ChromeCacheView від NirSoft. Універсальний застосунок для перегляду кешу всіх браузерів на базі платформи Chromium. Можна провести вибірку й прочитати файл за його реальним шляхом в файловій системі.

ChromeCookieViewer від NirSoft. Універсальний застосунок для перегляду кукі всіх браузерів на базі платформи Chromium. Доступний зміст кукі, дата його створення, доступу та запланованого видалення.

Далі розглянемо інструменти аналізу, які реалізуються як плагіни до браузера. Тобто збирають дані з середини. Їх перевага – кросплатформовість, відкритий код (незалежно від ліцензії).

WebHistorian: Education edition [14]. Плагін дозволяє переглядати сайти, пошукові запити та час відвідувань. Історія відвідувань демонструється за допомогою спеціального графіку, показаного на рис. 9. У кожне коло можна перейти за детальною інформацією й переглянути таблицю по відвідуванням сайту: сторінка, заголовок й час.

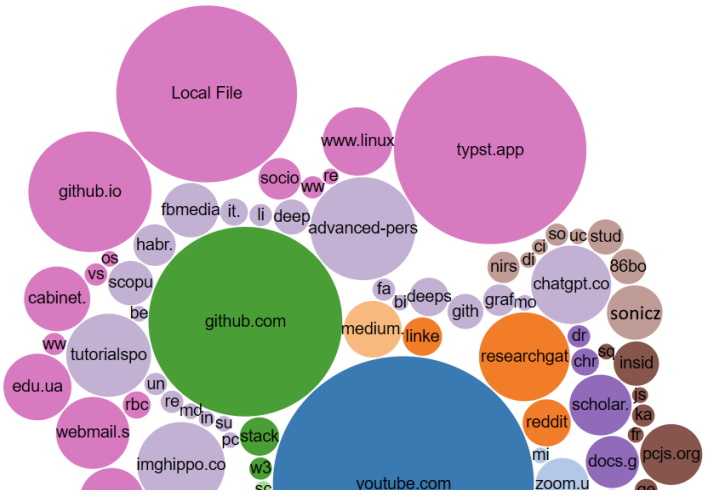

Рис. 9. Приклад візуалізації відвідуваності та сайтів за відвідуваністю (розмір) та категорією (колір)

Історія пошукових запитів візуалізується як хмара тегів за окремими словами. Але по кожному слову можна клацнути правою кнопкою миші й переглянути запити, які містили це ключове слово. Кольорова карта візуалізує активність користувача за годинами та днями тижня. Копія екрану показана на рис. 10. Знизу кольорова легенда показує яка приблизно кількість візитів спостерігається за годину. По кожному квадратику можна клацнути, та переглянути які конкретно сторінки були переглянуті в цей період часу.

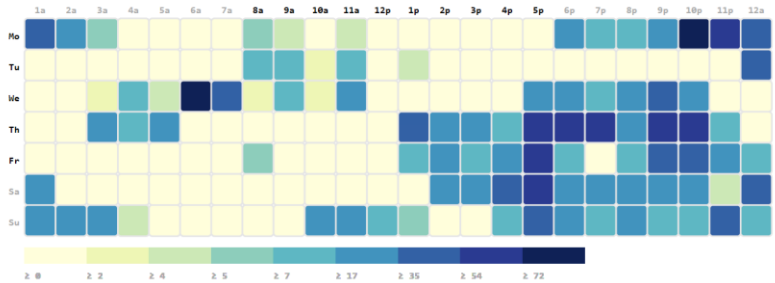

Рис. 10. Приклад візуалізації відвідуваності за годинами та днями тижня

History Trends Unlimited. Аналогічний до попереднього плагін [15], але менш зручний в користуванні, оскільки представляє дані просто як таблиці. Проте, в нього є можливість класифікації за типом переходу (пошуковий запит, пряме введення адреси, перехід з іншої адреси, згенероване ШІ посилання тощо) – рис. 11.

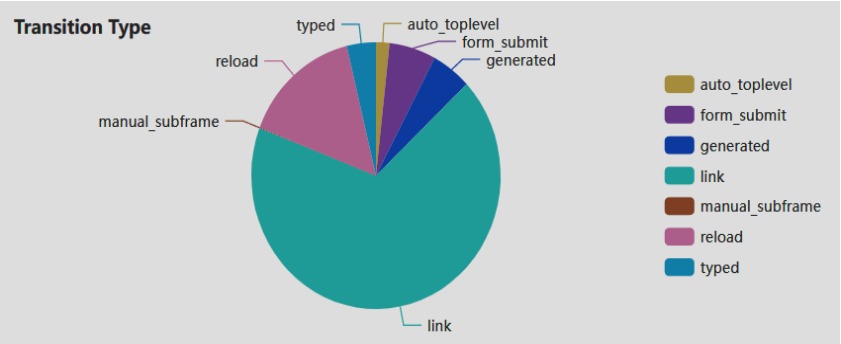

Рис. 11. Кругова діаграма за типами переходів [16].

Нарешті, розглянемо спеціалізовану програму для комплексного збору даних, яка називається Hindsight [17]. Ця програма реалізується в двох версія й по суті являє собою Python скрипт. В графічній версії інтерфейс реалізується як вебінтерфейс. В ньому можна задати шлях до профілю, запустити перегляд, переглянути загальну статистику та експортувати результати аналізу в Excel файл чи в JSON. Приклад загальних результатів аналізу показаний на рис. 12.

Detected Chrome version:	130-134	Login Data records:	202	DIPS Items:	0
URL records:	2570	Preference Items:	9582	Extension Rules records:	297
Download records:	63	Extensions:	21	Extension Scripts records:	32022
Cache records:	4920	Extension Cookie records:	2	Extension State records:	10512
GPU Cache records:	0	IndexedDB records:	9760	Local Extension Settings records:	57076
Cookie records:	6136	Session Storage records:	163928	Managed Extension Settings records:	0
Local Storage records:	28703	Site Characteristics records:	42812	Sync App Settings records:	0
Bookmark records:	52	File System Items:	495	Sync Extension Settings records:	465
Autofill records:	830	DIPS Popup Items:	32	HSTS records:	3961

Рис. 12. Статистика обробки профілю Hindsight

В порівнянні з попередніми інструментами цей інструмент найменш зручний, оскільки не має фактично графічного інтерфейсу, який допомагає аналізувати дані. Проте, Hindsight надає певну інформацію, яку не можливо отримати за допомогою інших інструментів. Це інформація про те, на яких сайтах застосовувалось автозаповнення форм (зокрема з паролями) та під яким логіном проводилась авторизація, це всі збережені дані у storage для вебзастосунків й це всі запити, які проводять періодично розширення. У цілому програма Hindsight – це гарна точка відліку для розробки застосунків з певними алгоритмами аналізу та візуалізації.

Висновки. Розглянуто аналіз збережених в браузері даних, які мають бути потрібні як докази при криміналістичному аналізі. Детально проаналізована загальна структура та формат збереження даних в профілі браузера. Описане застосування загальних інструментів криміналістичного аналізу й аналіз дампів. Проаналізоване застосування спеціалізованих засобів аналізу профілів браузерів на платформі Chromium, а саме такі.

1. Спеціалізовані утиліти для збору даних з позиції одного типу інформації (кеш, історія відвідувань, кукі тощо). Певна інформація вимагає закриття браузера, інша може бути зібрана й при відкритому браузері.
2. Комплексні засоби аналізу, реалізовані як плагіни для браузера. При цьому підході до реалізації системи не до всієї інформації наявний доступ, оскільки наявні обмеження браузера для плагінів, введені з метою безпеки й не тільки.
3. Комплексні засоби аналізу, яким не потрібен запуск в межах браузера. Ці інструменти здатні зібрати найбільшу кількість інформації. Проте звичайно працювати з нею не дуже зручно.

Список літератури

1. Mandine L. Browser Forensics. Medium. 2024. URL: https://medium.com/@laurent.mandine/browser-forensics-89429fe0749f

2. Кіберпсихологія у вимірах сучасного наукового дискурсу: монографія / С. Хаджирадєва, Ю. Левін, М. Тодорова; ред. С. К. Хаджирадєва. Одеса: Астропринт, 2024. 240 с.
3. The official GitHub of the Chromium. URL:https://github.com/chromium/chromium
4. Dušek D. Identification of specific Google Chrome user based on analysis of its application data. *Proc. of Studentská Konference Inovací, Technologií a Vědy v IT*. *Brno*. 2016. URL: https://excel.fit.vutbr.cz/submissions/2016/044/44.pdf
5. Shafqat N. Forensic investigation of user's web activity on Google Chrome using various forensic tools. *Int. J. Comput. Sci. Netw. Secur*. 2016. No.16(9). P.123–132.
6. Rathod D. Web browser forensics: Google Chrome. *Int. J. of Advanced Research in Computer Science*. 2017. V.8. No. 7. URL: https://ijarcs.info/index.php/Ijarcs/article/view/4433
7. Flowers C., Haider A. Web browser artefacts in private and portable modes: A forensic investigation. *International Journal of Electronic Security and Digital Forensics*. 2016. No.8. P.99–117. URL: https://doi.org/10.1504/IJESDF.2016.075583
8. Horsman G., Findlay B., Edwick J., Asquith A., Swannell K. A forensic examination of web browser privacy-modes. Forensic Science International: Reports. Volume 1. 2019. URL: https://doi.org/10.1016/j.fsir.2019.100036.
9. Hughes K., Papadopoulos P., Pitropakis N., Smales A. Private Mode: Is It What We Were Promised? *Computers.* 2021. No. 10. P.165. URL: https://doi.org/10.3390/computers10120165
10. Volatility plugin: Chrome History. URL: https://blog.superponible.com/2014/08/31/volatility-plugin-chrome-history/
11. Chrome history plugin for Volatility 3. URL: https://github.com/its-radio/volatility_plugins/blob /main/chrome_hx/chrome_hx.py
12. Choi G., Bang J., Lee S., Park J. Chracer: Memory analysis of Chromium-based browsers. *Forensic Science International: Digital Investigation.* 2023. Vol. 46. DOI: 10.1016/j.fsidi.2023.301613.
13. Unique collection of small and useful freeware utilities. URL: nirsoft. net
14. Digital Forensic Analysis of Web Browser Records. URL: webhistorian.com
15. History Trends Unlimited https://chromewebstore.google.com/detail/history-trends-unlimited/pnmchffiealhkdloeffcdnbgdnedheme
16. Chromium is an open-source browser project. URL: https://chromium.googlesource.com/chromium/src/+/master/chrome/common/extensions/api/history.json
17. Hindsight: Internet history forensics for Google Chrome/Chromium. URL: https://github.com/obsidianforensics/hindsight

WEB BROWSER HISTORY FORENSIC ANALYSIS

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 Str., Odesa, 65023, Ukraine
Email: stopakevich@gmail.com

The present study analyzes the capabilities of existing software tools for the forensic analysis of the history of the most commonly used browsers based on the Chromium platform (Chrome, Edge, Opera, etc.) to determine their effectiveness. The existing scheme for storing user data and actions in the Edge profile has been formalized and analyzed, and the locations where various settings and files of this browser are stored have been identified. A thorough analysis of the Chromium source code and the actual data stored by the Edge browser was conducted to elucidate the relationships and interconnections between the entities of the SQLite database of the profile and the contents of its main JSON file. The existing software that facilitates forensic analysis of profile data is reviewed. The following are utilities for working with data and dumps, specialized utilities for collecting data from a browser profile from the perspective of one type of information—cache, visit history, cookies—and more complex tools for analyzing browser profiles. The latter category facilitates the collection of a more substantial amount of information. However, in general, the potential exists for significant improvement in the realm of complex software, given its present suboptimal functionality, particularly from the perspective of forensic experts who utilize it.

Keywords: forensic analysis, browser, data collection, browser plugins, activity analysis, malware, criminal activity timeline.