УДК 004

УДОСКОНАЛЕНИЙ МЕТОД ВИЯВЛЕННЯ ВЕБ-СКРАПЕРІВ З ВИКОРИСТАННЯМ ПАСТОК

П. П. Саханда[1, a], В. М. Ткач[1]

[1] *Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Фізико-технічний інститут*

Анотація

Здійснено опис методів виявлення та блокування веб-скраперів, а саме: використання файлів *cookie* та мови програмування *JavaScript*, обмеження кількості запитів до веб-серверу, обфускація даних та використання пасток. Досліджено та вдосконалено метод виявлення веб скраперів з використанням пасток. Зроблено висновок про ефективність даного методу.

Ключові слова: веб-скрапер, пастка, виявлення, автоматизований доступ.

Вступ

Веб-скрапінг – це збір даних з різноманітних інтернет-ресурсів. Загальний принцип його роботи можна пояснити наступним чином: певний автоматизований код виконує GET-запити на сервер, отримує відповідь, аналізує $HTML$-документ, шукає потрібну інформацію та перетворює її в попередньо вибраний формат.

У свою чергу, веб-скрапер – це програма, спроектована для автоматичного збору даних з інтернету та подальшої їх обробки. Тобто, веб-скрапери дозволяють автоматизувати доступ до веб сайтів, імітуючи при цьому поведінку людини. Ці інструменти дозволяють автоматично отримувати нові чи оновлені дані та зберігати їх для подальшої обробки та використання.

1. Архітектура веб-скрапера та сфери застосування

В наш час веб-скрапінг використовують з багатьох причин: агресивна конкуренція, Інтернет-перегони, шахрайство, хакерські атаки та спам. Веб-скрапінг також використовують, щоб без особливих зусиль викрасти будь-які потрібні дані. Вони часто імітують звичайну поведінку користувачів, що ускладнює їх виявлення та блокування.

Архітектура веб-скраперів

Зазвичай веб-скрапери працюють за наступним алгоритмом:

1) Підготовка механізму отримання $HTML$-коду по запиту типу GET.
2) Аналіз DOM-структури потрібного інтернет-ресурсу.
3) Визначення вузлів з потрібною інформацією.
4) Налаштування обробника вузлів.
5) Виведення даних в нормалізованому виді (наприклад, в форматі $JSON$).

Схема роботи показана на рисунку 1.

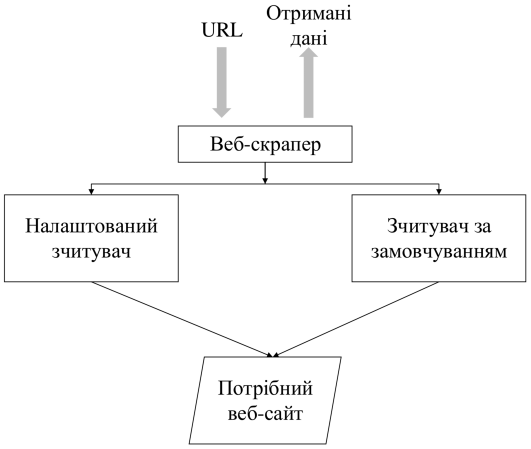

Рис. 1. Схема роботи веб-скрапера

На вході система отримує URL потрібної сторінки, а на виході віддає нормалізовані дані (наприклад, в форматі $JSON$). Отримавши URL, система визначає, якому зчитувачу потрібно направити сторінку на обробку. У випадку, якщо система знає архітектуру веб-ресурсу, URL отримує спеціально налаштований зчитувач, в іншому випадку – сторінку починає аналізувати зчитувач, який використовується за замовчуванням. Як правило, в таких випадках використовується найбільш стабільний зчитувач [1].

Сфери використання веб-скраперів

Можливі сценарії використання інструментів веб-скрапінгу:

- Збір даних для маркетингових досліджень;

[a] psakhanda@gmail.com

- Збір даних з інтернет-ресурсів, що мають інформаційний характер (новинні ресурси, електронні видання, газети та журнали);
- Створення баз даних на основі відкритих контактних даних (електронні адреси користувачів, номери телефонів);
- Збір даних з інтернет-магазинів для моніторингу цін у власних цілях;
- Викрадення авторського контенту для подальшого використання (статті, зображення, відео);
- Збір інформації про спеціальні пропозиції та знижки;
- Збір інформації з порталів для продажу нерухомості;
- Інтеграція даних з декількох джерел;
- Моніторинг сайтів для бронювання квитків, готелів;
- Збір урядових даних.

Якщо деякі з наведених сфер використання веб-скраперів не є критичними, то інші можуть завдати серйозних збитків компаніям, виходити за рамки етики, моралі та, в окремих випадках, навіть законів.

2. Методи виявлення та блокування веб-скраперів

Є достатньо багато способів, за допомогою яких можна виявити веб-скрапер чи заблокувати його роботу. В ході дослідження було розглянуто найбільш поширені та ефективні методи. Нижче наведено їх короткий опис:

- Використання файлів *cookie* та мови програмування *JavaScript*. Ці засоби використовують для того, щоб переконатися, що користувач відвідує сайт через звичайний веб-переглядач. Більшість простих інструментів для веб-скрапінгу не можуть обробляти складний *JavaScript* код або зберігати файли *cookie* [2].
- Обмеження кількості запитів до серверу. Автоматизовані клієнти зазвичай надсилають запити до веб-сторінок набагато частіше, ніж реальні користувачі. Більш удосконалені веб-скрапінгові системи намагаються уникнути виявлення, зменшуючи швидкість та періодичність запитів. Проте, автоматизовані системи в будь-якому разі будуть виявляти себе, звертаючись до веб-сторінки після відповідного проміжку часу [3].
- Обфускація даних. Веб-скрапери розроблені для вилучення тексту з веб-сторінок. Відображення веб-сторінок у вигляді зображень або флеш-файлів може перешкоджати веб-скраперам отримати потрібну інформацію [3].
- Використання пасток. Цей метод буде детально розглянуто в наступному розділі.

3. Використання пасток для виявлення веб-скраперів

Пастки – потрібний і корисний інструмент для виявлення і запобігання збору інформації веб-скраперами та веб-роботами. При потраплянні такої системи в пастку, можна записувати IP-адреси та обмежувати і блокувати для них доступ до інтернет-ресурсів. Хоча комерційні системи захисту і не публікують технічні деталі про свої методи виявлення веб скраперів, можна припустити, що такі методи цілком можливо можуть використовуватись [4].

Види пасток

Пастка – популярний метод виявлення «небажаних» відвідувачів на сайтах. Існує два види пасток: активний і пасивний. Активні пастки намагаються заманити в пастку небажаний трафік. Пасивні пастки використовуються для виявлення веб-роботів, скраперів та ботів.

Одним з популярних видів активних пасток є «пастка для павуків», яка заманює веб-сканери в нескінченні цикли запитів на сторінках веб-сайту. «Пастка для павуків» може бути створена декількома способами, наприклад, шляхом додавання посилань з нескінченно глибокими деревами каталогів, або з допомогою динамічних сторінок з необмеженими параметрами. Однак недоліком таких активних пасток являється те, що в такі «пастки для павуків» можуть потрапити і дозволені веб роботи (наприклад, *Google* роботи) [5].

Прикладом пасивної пастки можна взяти приховування посилань та інших ресурсів на веб-сторінці з використанням форматування. Звичайні користувачі, що користуються браузерами для перегляду веб-сторінок, не можуть бачити посилання чи інші ресурси, які були попереднього приховані від перегляду за допомогою можливостей форматування елементів сторінки (мова CSS).

Звичайно, існують веб-роботи, які можуть обходити деякі методи захисту на основі пасток. Удосконалені веб-роботи можуть використовувати *cookie*, виконувати код *JavaScript*, емулювати клавіатуру і комп'ютерну мишку, аналізувати код та відвідувати сайт лише по попередньо визначених після аналізу директоріях [5].

Удосконалення методу виявлення веб-скраперів з використанням пасток

Все частіше можна знайти веб-скрапери, що можуть обходити стандартні методи захисту. Наприклад, зустрічаються сканери, які можуть бути налаштовані на визначені директорії. Саме для таких випадків в ході досліджено було запропоновано вдосконалений метод виявлення веб-скраперів з використанням пасток.

Удосконалення методу полягає в покритті всіх директорій сайту пастками без додаткового створення фізичних сторінок-пасток. Було розроблено скрипт, який автоматично додає до кожного $HTML$-файлу на сайті прихований для звичайного користувача блок з посиланням на сторінку-пастку. Посилання вказує на пастку, яка знаходиться саме в тій директорії, де знаходиться відповідний $HTML$-файл. Таким чином, якщо веб-скрапер опрацьовує лише ті дире-

кторії сайту, які йому потрібні, він в будь-якому разі потрапить в одну з пасток.

При підготовці до застосування даного методу було створено лише одну фізичну сторінку-пасту в кореневому каталозі веб-сайту. Також було розроблено скрипт, який генерує вміст файлу *.htaccess*. Приклад *.htaccess* файлу для такого методу додавання пасток на сторінки веб-сайту показаний на рисунку 2.

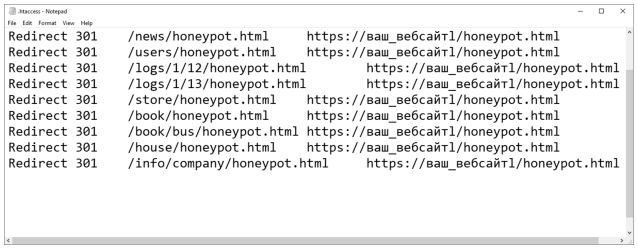

Рис. 2. Приклад *.htaccess* файлу

В файлі *.htaccess* зазначається кожна віртуальна пастка (їх кількість дорівнює кількості каталогів, створених на сайті) і перенаправляється на попередньо створену фізичну пастку. При потраплянні на цю сторінку зчитується IP-адреса користувача чи системи, що надсилає запит до сервера. Схема роботи такої пастки продемонстрована на рисунку 3. Для прикладу взято одне з запропонованих на рисунку 2 перенаправлень.

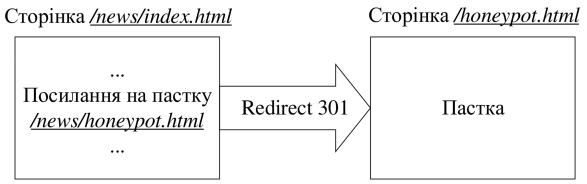

Рис. 3. Схема роботи запропонованого методу представлення пасток

Даний метод забезпечує стовідсоткове покриття веб-сайту пастками і за рахунок цього дозволяє виявити більш удосконалені веб-скрапери. Таким чином вирішується проблема можливого пропуску веб-скрапером пастки, створеної на веб-сайті.

4. Висновок

У даній статті було розглянуто принципи роботи веб-скраперів та сфери їх використання, проведено аналіз деяких методів виявлення та блокування веб-скраперів. Також було детально розглянуто метод виявлення веб-скраперів з використанням пасток та запропоновано вдосконалення даного методу.

Ефективність методу полягає в виявленні тих веб-скраперів, що сканують веб-сайт лише по певних директоріях. Досить часто пастки створюються в кореневому каталозі, а веб-скрапери в свою чергу пропускають такі місця, і сканують певні директорії, що знаходяться глибше. Такі веб-скрапери не реагують на посилання, що ведуть до інших директорій чи до кореневого каталогу. В таких випадках виявити присутність веб-скрапера на сайті, використовуючи звичайний механізм додавання пасток, неможливо.

Перелік використаних джерел

1. Olston, C. and Najork, M. Web crawling. 2010. — *Foundations and Trends in Information Retrieval* 3:175-246.
2. Doran, D and Gokhale, S. S. Web robot detection techniques: overview and limitations. 2011. — *Data Mining and Knowledge Discovery* 22:183-210.
3. Doran, D. Detection, Classification, and Workload Analysis of Web Robots. — 2014. — Available at: `http://digitalcommons.uconn.edu/dissertations/348`.
4. Dryer, A. J. and Stockton, J. Internet 'Data Scraping': A Primer for Counseling Clients. 2013. — *New York Law Journal* 15:1-3.
5. Mitchell, R. Web Scraping with Python: Collecting Data from the Modern Web. S. : O'Reilly Media, 2015.