НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»
УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ М.В.Грайворонський

“___”_____2019 р

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека

на тему: Система виявлення веб-скраперів з використанням пасток

Виконав: студент 2 курсу, групи ФБ-71мн
(шифр групи)

Саханда Павло Петрович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.е.н., доцент каф. ІБ Ткач В.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою
Спеціальність (спеціалізація) – 125 Кібербезпека («Системи, технології та математичні методи кібербезпеки)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Саханді Павлу Петровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Система виявлення веб-скраперів з використанням пасток,
науковий керівник дисертації Ткач Володимир Миколайович, к.е.н., доцент каф. ІБ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «02» квітня 2019 р. № 1023-с

2. Термін подання студентом дисертації 06.05.2019 р.

3. Об'єкт дослідження веб-сайти та інтернет-ресурси, вразливі до досліджуваної атаки, та можливі методи захисту від неї

4. Предмет дослідження виявлення веб-скраперів за допомогою пасток, розміщених на веб-сайті

5. Перелік завдань, які потрібно розробити аналіз роботи веб-скраперів; аналіз можливих загроз для власників веб-сайтів при відсутності захисту від веб-скраперів; вдосконалення існуючих методів захисту; створення власної системи виявлення веб-скраперів на основі вдосконаленого методу; побудова програмної моделі веб-сайту з використанням запропонованої системи виявлення; оцінка захищеної моделі та аналіз результатів.

6. Орієнтовний перелік ілюстративного матеріалу схема роботи веб-скраперів, схема роботи вдосконаленого методу виявлення веб-скраперів з використанням пасток, архітектура системи виявлення веб-скраперів, схеми роботи кожної з підсистем системи виявлення веб-скраперів

7. Орієнтовний перелік публікацій «Вдосконалений метод виявлення веб-скраперів з використанням пасток»

8. Дата видачі завдання 07.09.2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вибір тематики роботи	09.2018-10.2018	виконано
2.	Опрацювання необхідної літератури	11.2018	виконано
3.	Складання структури роботи	11.2018-12.2018	виконано
4.	Проведення необхідних досліджень	01.2019	виконано
5.	Проведення необхідних експериментів	01.2019	виконано
6.	Розробка архітектури системи виявлення	02.2019	виконано
7.	Розробка програмної моделі	02.2019-03.2019	виконано
8.	Проведення експериментів та аналіз отриманих результатів	04.2019	виконано

Студент _____ Саханда П.П.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Ткач В.М.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Представлена робота обсягом 87 сторінок містить 26 ілюстрацій, 12 таблиць та 17 джерел за переліком посилань.

Актуальність роботи зумовлюється тим, що на даний момент при розробці багатьох інтернет-ресурсів та веб-сайтів веб-розробники та замовники не завжди задумуються про правильне використання своїх ресурсів та захист даних, що знаходяться у відкритому доступі. Перш за все потрібно мінімізувати автоматизований доступ до таких даних та ефективно використовувати наявні ресурси для захисту контенту, особистих даних, що знаходяться у відкритому вигляді, чи інформації, що є об’єктом захисту авторського права.

Метою роботи є підвищення рівня захищеності інформації, що розміщена на інтернет-ресурсах різного характеру, шляхом створення ефективної системи, яка буде виявляти та автоматично блокувати виявлене програмне забезпечення для автоматизованого збору інформації з веб-сайтів.

Для досягнення даної мети були поставлені наступні завдання:

- аналіз роботи веб-скраперів;
- аналіз можливих загроз для власників веб-сайтів при відсутності захисту від веб-скраперів;
- визначення найефективніших методів захисту;
- вдосконалення існуючих методів захисту;
- створення власної системи виявлення веб-скраперів на основі вдосконаленого існуючого методу виявлення;
- побудова програмної моделі веб-сайту з використанням запропонованої системи виявлення веб-скраперів;
- спроба реалізації атаки на побудовану модель веб-сайту з використанням створеної системи;
- оцінка захищеної моделі та аналіз результатів.

Об’єктами дослідження є веб-сайти та інтернет-ресурси, вразливі до досліджуваної атаки, та можливі методи захисту від неї.

Предметом дослідження є виявлення веб-скраперів за допомогою пасток, розміщених на веб-сайті.

Методами дослідження було обрано: опрацювання літератури за даною темою, аналіз причин виникнення даної атаки, аналіз методів захисту, проведення експериментів та порівняння деяких методів.

Наукова новизна. В ході проведення дослідження було вперше побудовано систему виявлення веб-скраперів, яка є ефективною при виявленні програмного забезпечення під час його запуску лише в певних директоріях веб-сайту. При цьому система не потребує створення додаткових файлів, які будуть завантажувати сервер.

Практичне значення полягає в тому, що результати роботи можуть застосовуватись при створенні інтернет-ресурсів для мінімізації можливості крадіжки контенту, особистих даних, що знаходяться у відкритому вигляді, чи інформації, що є об’єктом захисту авторського права.

Результати дослідження, які є основою магістерської дисертації, були опубліковані в статті «Удосконалений метод виявлення веб-скраперів з використанням пасток». Стаття опублікована в збірнику, що містить матеріали XVII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики».

Ключові слова: веб-скрапер, система виявлення, пастка, веб-сайт, приватність, безпека, автоматизований доступ.

ABSTRACT

This work of 87 pages contains 26 illustrations, 12 tables and 17 literature references.

The relevance of the work is due to the fact that at the moment many web developers and customers do not always think about the proper use of their resources and protection of data that is in the open access, when they develop Internet resources and websites. First of all, it is necessary to minimize automated access to such data and to make effective use of available resources to protect content, personal data that is in the public access, or information that is the subject of copyright protection.

The aim of the work is to increase the security of information placed on various resources of the Internet by creating an effective system that will detect and automatically block the detected software for automated information gathering from websites.

To achieve this goal, the following tasks were set:

- analysis of the work of web scrapers;
- analysis of possible threats to website owners in the absence of protection from web scrapers;
- the most effective protection methods identifying;
- existing protection methods improvement;
- designing of own web scrapers detection system, based on an improved existing detection method;
- creating a program model of a website, using the proposed web scrapers detection system;
- an attempt to implement an attack on a built-in website model, using the created system;
- evaluation of the protected model and analysis of the results.

The objects of the study are websites and Internet resources, vulnerable to the attack being investigated and possible protection methods against it.

The subject of the study is the detection of web scrapers, using honeypots, located on the website.

The research methods were chosen: studying the literature on this topic, analyzing the causes of this attack, analyzing protection methods, conducting experiments and comparing some methods.

Scientific novelty. During the study, a web scrapers detection system, which is effective in detecting software, when it is launched only in certain directories of the website, was built for the first time. The system does not require additional files that will load the server.

The practical value is that work results can be applied when creating Internet resources to minimize the possibility of theft of content, personal data that is in the public access, or information that is the subject of copyright protection.

The research findings, which are the basis of the master's thesis, were published in the article "Improved web scrapers detecting method using honeypots". The article is published in the collection containing materials of the XVII All-Ukrainian Scientific and Practical Conference of Students, Aspirants and Young Scientists "Theoretical and Applied Problems of Physics, Mathematics and Computer Science".

Keywords: web scraper, detection system, honeypot, website, privacy, security, automated access.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

DOM – Document Object Model.

GET-запит – запит, що виконується для отримання вмісту вказаного ресурсу.

URL – Uniform Resource Locator.

SEO – Search Engine Optimization.

.htaccess – конфігураційний файл веб-сервера Apache, який дозволяє керувати роботою веб-сервера.

CAPTCHA – completely automated public turing test to tell computers and humans apart.

CSS – Cascading Style Sheets

ПЗ – програмне забезпечення

ВСТУП

Загроза несанкціонованого збору інформації з веб-ресурсів в даний час є дуже актуальною. З кожним роком кількість трафіку, класифікованого як веб-скрапінг, згідно з дослідженнями, значно збільшується.

З кожним днем поліпшуються методи і механізми захисту веб-роботів від виявлення.

Найбільш популярними цілями для веб-скрапінгу є сайти для продажу квитків, урядові портали, новинні ресурси, електронні газети та журнали, сайти для розміщення оголошень в сфері подорожей і нерухомості, а також соціальні мережі.

Саме через це створення системи виявлення веб-скраперів є досить актуальним питанням в наш час. Так само як вдосконалюються веб-скрапери мають вдосконалюватися і методи їх виявлення.

Актуальність роботи зумовлюється тим, що на даний момент при розробці багатьох інтернет-ресурсів та веб-сайтів веб-розробники та замовники не завжди задумуються про правильне використання своїх ресурсів та захист даних, що знаходяться у відкритому доступі. Перш за все потрібно мінімізувати автоматизований доступ до таких даних та ефективно використовувати наявні ресурси для захисту контенту, особистих даних, що знаходяться у відкритому вигляді, чи інформації, що є об’єктом захисту авторського права.

Мета роботи – підвищити рівень захищеності інформації, що розміщена на інтернет-ресурсах різного характеру, шляхом створення ефективної системи, яка буде виявляти та автоматично блокувати виявлене програмне забезпечення для автоматизованого збору інформації з веб-сайтів.

Для досягнення даної мети було поставлено наступні **завдання:**

- аналіз роботи веб-скраперів;

- аналіз можливих загроз для власників веб-сайтів при відсутності захисту від веб-скраперів;
- визначення найефективніших методів захисту;
- вдосконалення існуючих методів захисту;
- створення власної системи виявлення веб-скраперів на основі вдосконаленого існуючого методу виявлення;
- побудова програмної моделі веб-сайту з використанням запропонованої системи виявлення веб-скраперів;
- спроба реалізації атаки на побудовану модель веб-сайту з використанням створеної системи;
- оцінка захищеної моделі та аналіз результатів.

Об’єктами дослідження є веб-сайти та інтернет-ресурси, вразливі до досліджуваної атаки та можливі методи захисту від неї.

Предметом дослідження є виявлення веб-скраперів за допомогою пасток розміщених на веб-сайті.

Методами дослідження було обрано: опрацювання літератури за даною темою, аналіз причин виникнення даної атаки, аналіз методів захисту, проведення експериментів та порівняння деяких методів.

Наукова новизна. В ході проведення дослідження було вперше побудовано систему виявлення веб-скраперів, яка є ефективною при виявленні програмного забезпечення під час його запуску лише в певних директоріях веб-сайту. При цьому система не потребує створення додаткових файлів, які будуть завантажувати сервер.

Практичне значення полягає в тому, що результати роботи можуть застосовуватись при створенні інтернет-ресурсів для мінімізації можливості крадіжки контенту, особистих даних, що знаходяться у відкритому вигляді, чи інформації, що є об’єктом захисту авторського права.

Результати дослідження, які є основою магістерської дисертації були опубліковані в статті «Удосконалений метод виявлення веб-скраперів з

використанням пасток». Стаття опублікована в збірнику, що містить матеріали XVII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики».

1 ВЕБ-СКРАПЕРИ ТА АНАЛІЗ ЇХ РОБОТИ

Веб-скрапінг – це автоматизований 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].

1.1 Передумови використання веб-скраперів

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.

Можливі сценарії використання інструментів веб-скрапінгу:

- Збір даних для маркетингових досліджень;
- 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;
- Збір інформації з порталів для пошуку роботи;
- Інтеграція даних з декількох джерел;
- Моніторинг сайтів для бронювання квитків, готелів;
- Моніторинг даних погоди;
- Збір урядових даних [2].

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.

1.2 Архітектура веб-скраперів та особливості їх роботи

Веб-скрапінг складається з наступних основних, тісно зв’язаних етапів: аналіз веб-сайтів, сканування веб-сайтів та організація і структуризація даних (див. рис. 1.1).

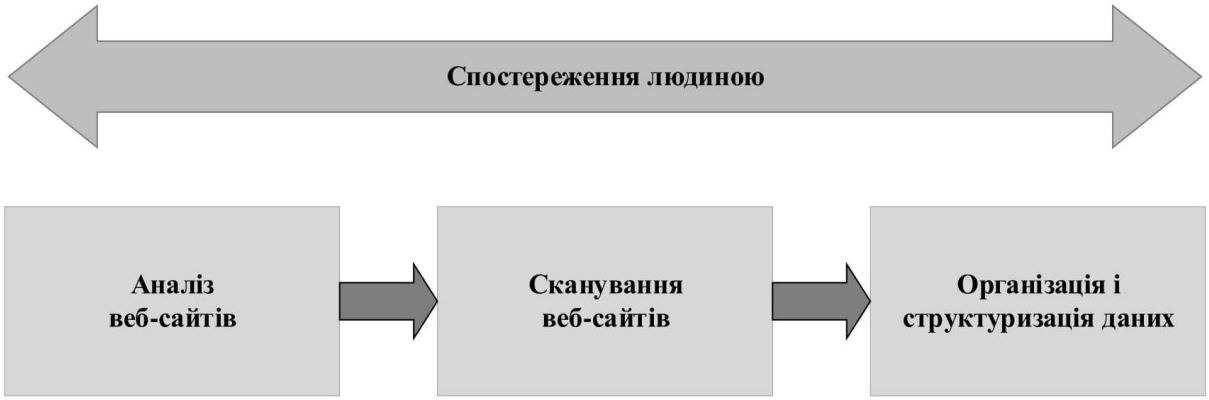

Рисунок 1.1 – Етапи веб-скрапінгу

Аналіз веб-сайту вимагає вивчення базової структури веб-сайту для розуміння того, як зберігаються, використовуються та опрацьовуються необхідні дані. Це вимагає базового розуміння архітектури мережі Інтернет, мови розмітки веб-сторінок (наприклад, HTML, CSS, XML, XBRL тощо) та базові розуміння баз даних (наприклад, MySQL).

Сканування веб-сайтів передбачає розробку та запуск скриптів, які автоматично переглядають веб-сайт і отримують необхідні дані. Ці програми сканування (або скрипти) часто розробляються з використанням таких мов програмування, як R і Python. Це пов’язано з загальною популярністю цих мов у бібліотеках Data Science і доступності (наприклад, «rvest» пакет в мові програмування R або Beautiful Soup library в мові програмування Python) для автоматичного сканування і аналізу веб-даних. Після того, як необхідні дані будуть проаналізовані з обраного ресурсу, їх необхідно очистити, попередньо обробити і організувати таким чином, щоб забезпечити подальший аналіз цих даних. Враховуючи обсяг даних, що використовується, програмний підхід може бути необхідним і для економії часу. Багато мов програмування, такі як R і Python, містять бібліотеки обробки природних мов (NLP) і функції обробки даних, корисні для очищення та організації даних.

Зазвичай ці три елементи веб-скрапінгу не можуть бути повністю автоматизованими і часто вимагають принаймні деякого втручання людини та контролю.

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 вигляді (наприклад, в форматі JSON).

Схема роботи показана на рисунку 1.2.

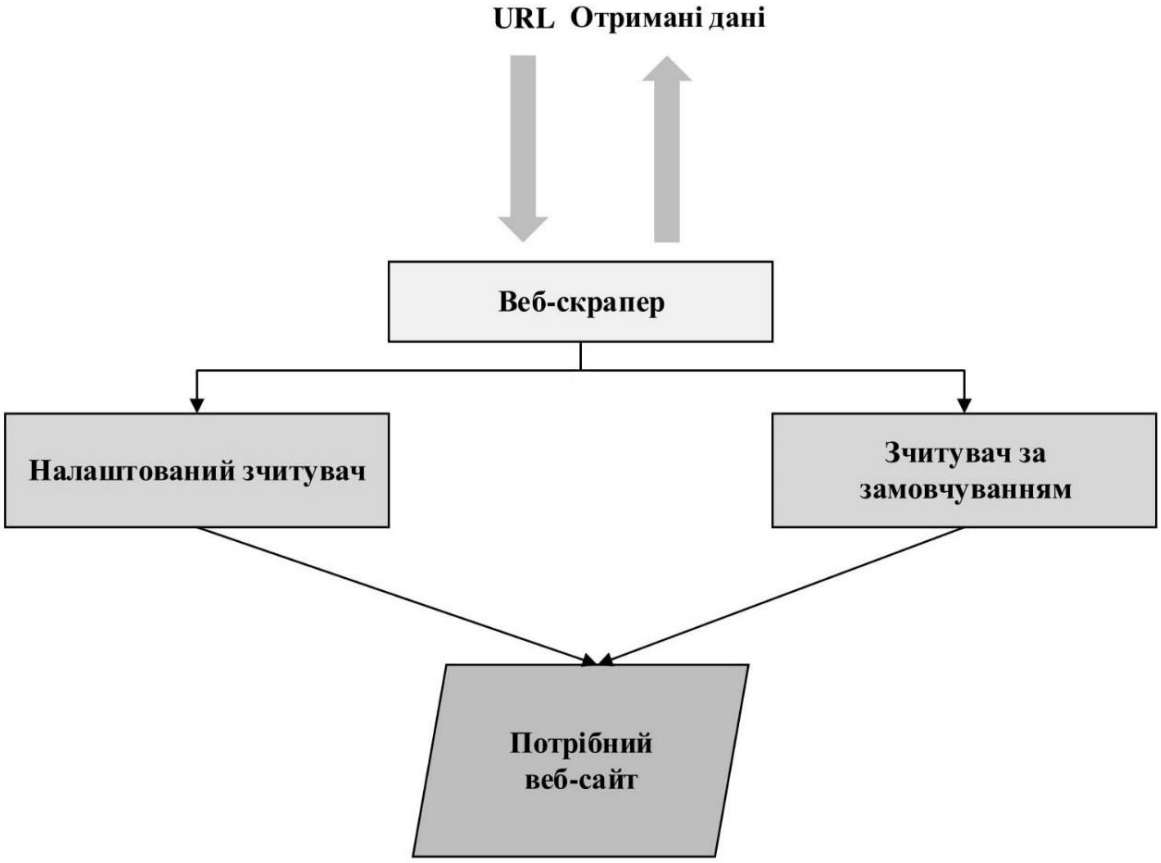

Рисунок 1.2 – Схема роботи веб-скрапера

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 [3].

1.3 Види веб-скраперів

Веб-скрапінг – це процес автоматичного збору даних або інформації в мережі Інтернет. Це сфера, що активно розвивається і досить часто вимагає певних удосконалень в автоматизованій обробці тексту, смисловому розумінні, сфері штучного інтелекту та взаємодіях типу «людина – комп'ютер».

Сучасні рішення для веб-скрапінгу варіюються від спеціальних, що вимагають людських зусиль, до повністю автоматизованих систем, які можуть перетворювати цілі веб-сайти в структуровану інформацію з деякими обмеженнями.

В цьому підрозділі будуть розглянуті можливі техніки веб-скрапінгу та види програмного забезпечення, що використовуються для реалізації атак.

1.3.1 Техніки веб-скрапінгу

Нижче наведено перелік можливих рішень, що можуть активно використовуватись для веб-скрапінгу:

1. Ручне копіювання та вставка інформації.

Іноді навіть найкраща автоматизована технологія збору інформації з веб-сторінок не може замінити аналіз зроблений людиною та ручне копіювання даних. В деяких випадках це може бути єдиним робочим та надійним рішенням, коли сайти цілеспрямовано створюють різні перепони для веб-роботів щоб запобігти автоматизованому доступу до ресурсів.

2. Відповідність текстового шаблону.

Простий та ефективний метод веб-скрапінгу для збору даних з веб-сторінок. Цей метод може використовувати команду grep в UNIX системах або пошук за допомогою регулярних виразів таких мов програмування як Perl чи Python.

3. HTTP-програмування.

Статичні та динамічні веб-сторінки можуть бути отримані шляхом посилання HTTP-запитів до віддаленого веб-сервера за допомогою мережевого програмування.

4. Синтаксичний аналіз HTML.

Багато веб-сайтів мають великі кількості сторінок, що динамічно генеруються із базового структурованого джерела, подібного до бази даних. Дані однієї категорії зазвичай кодуються на подібні сторінки звичайним сценарієм або шаблоном.

Під час інтелектуального аналізу даних програма, яка виявляє такі шаблони в певному джерелі інформації, витягує її вміст і переводить її у реляційну форму, що називається обгорткою. Алгоритми генерації обгорток припускають, що вхідні сторінки відповідають загальному шаблону і що їх можна легко ідентифікувати за загальною схемою URL.

Більш того, деякі напівструктуровані мови запитів, такі як Xquery і HTQL, можуть бути використані для розбору HTML-сторінок і для отримання, перетворення і систематизації вмісту сторінки.

5. Аналіз DOM.

Вбудовуючи повноцінний веб-браузер, наприклад, Safari або браузер Google Chrome, програми можуть отримувати динамічний вміст, створений скриптами на клієнтській стороні. Ці елементи керування веб-браузером також аналізують вміст веб-сторінки та його структуру. На основі таких даних програми можуть отримувати частини сторінок.

6. Вертикальна агрегація.

Деякі компанії займаються розробкою вертикально-специфічних платформ для збору інформації в мережі інтернет. Ці платформи створюють і контролюють безліч «ботів» для таких цілей без «людини в циклі» (без прямого залучення людини). Також такі платформи не потребують виконання робіт пов’язаних з конкретним сайтом.

Підготовка передбачає створення бази знань для використання у будь яких випадках, а потім така платформа автоматично створює ботів для аналізу інтернет ресурсів. Надійність та ефективність платформи вимірюється якістю інформації, яку вона отримує (як правило, кількістю полів) і її масштабованістю (наскільки швидко вона може масштабуватися до сотень або тисяч сайтів). Ця масштабованість в основному використовується для аналізу динамічних сайтів, які постійно збільшуються.

В основному звичайні агрегатори вважають такі сайти занадто складними і потребують докладання чималих зусиль для аналізу, обробки та збору інформації [4].

7. Розпізнавання семантичних анотацій.

Сторінки, на яких здійснюється сканування, можуть охоплювати метадані або семантичні розмітки і анотації, які можна використовувати для пошуку окремих фрагментів даних. Якщо анотації вбудовані на сторінках, як це робить Microformat, цю техніку можна розглядати як особливий випадок розбору DOM. В іншому випадку, анотації, організовані в семантичний шар, зберігаються і керуються окремо від веб-сторінок, тому веб-скрапери можуть отримати схему даних і інструкції з цього шару перед тим, як збирати дані зі сторінки.

8. Аналіз веб-сторінок за допомогою комп’ютерного зору.

Існують деякі методи, які використовують машинне навчання та комп’ютерне бачення, які намагаються ідентифікувати та витягувати інформацію з веб-сторінок, інтерпретуючи сторінки візуально, як це робить людина.

9. Xpath.

XML Path Language, або Xpath – це мова запитів, що працює з XML-документами. Оскільки документи XML основані на представленні документів у вигляді дерев, Xpath можна використовувати для навігації по дереву, вибираючи вузли на основі різних параметрів. Xpath можна

використовувати спільно з аналізом DOM і витягувати всю веб-сторінку, а також публікувати її в потрібному місці.

10. Засоби Google документів.

Google таблиці можна використовувати як інструмент веб-скрапінгу, і він дуже популярний серед користувачів. З допомогою Google таблиць, веб-скрапер може використовувати функцію IMPORTXML(,), щоб витягувати дані з веб-сайтів. Цей метод є доволі ефективним в випадках, коли за допомогою методів веб-скрапінгу користувач хоче отримати конкретні дані чи шаблони з веб-сайту. Також цей метод використовують для перевірки чи захищений веб-сайт від веб-скрапінгу [5].

1.3.2 Інструменти веб-скрапінгу

Оскільки існують різні інструменти веб-скрапінгу та послуги, що пропонують компанії, дуже важко визначити які з них є більш ефективними та надійніші в порівнянні з іншими. Основуючись на тому, як вони працюють, нижче наведено класифікацію інструментів веб-скрапінгу, доступних на ринку (див. рис. 1.3):

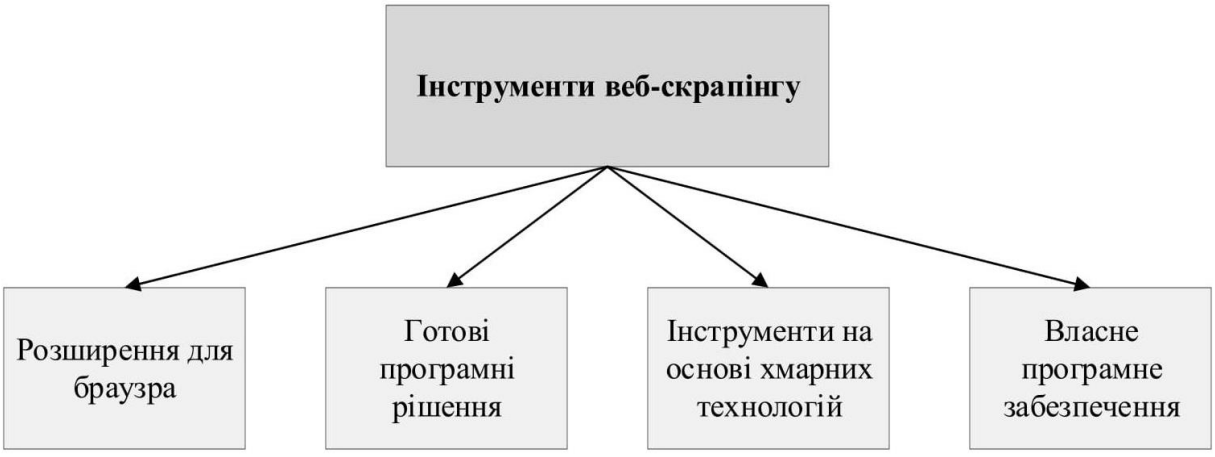

Рисунок 1.3 – Класифікація інструментів веб-скрапінгу

1. Розширення для браузера.

Розширення для браузера є прекрасним інструментом, щоб витягти невеликі частини даних. Веб-скрапінг за допомогою плагіна для браузера, а не окремого програмного забезпечення, встановленого на ПК є найбільш ефективним рішенням для перегляду, аналізу та подальшої модифікації даних.

Розширення для браузера дозволяє встановити його і вибрати спосіб, за допомогою якого потрібно отримати дані з вибраного веб-сайту. Дані буде завантажено у вибраному форматі (наприклад CSV, JSON) або в будь-якому іншому зручному форматі. Розширення для браузера майже не вимагають зусиль для встановлення та налаштування. Такі розширення мають лише мінімальні налаштування для отримання даних з веб-сайтів.

Також, через свою простоту, такі рішення мають певні обмеження. Розширення для браузера може витягти дані лише з однієї сторінки за один раз. Для обробки великої кількості даних є інші типи інструментів для веб-скрапінгу.

2. Готові програмні рішення.

Оскільки попит на дані та інформацію зростає з кожним днем, IT компанії взялися за розробку повноцінного програмного забезпечення для веб-скрапінгу.

Як і будь-яке інше програмне забезпечення, програмне забезпечення для веб-скрапінгу передбачає його повноцінне встановлення на ПК. Більшість таких програм повністю сумісні з найбільш поширеними операційними системами (Windows, MacOS).

Таке програмне забезпечення має більш широкий спектр налаштувань, тому таке рішення може бути більш гнучким і ефективним. Для початку роботи достатньо лише підлаштувати веб-скрапер під свої потреби.

Повноцінні програми для веб-скрапінгу використовують найбільш популярні і зручні у подальшому використанні формати для зберігання зібраних даних, такі як JSON, CSV.

На відміну від розширень для браузерів, програми для ПК дозволяють одночасно сканувати декілька сторінок і можуть обробляти більшу кількість інформації. Вони розраховані на невеликі та середні сайти.

3. Інструменти на основі хмарних технологій

У порівнянні з іншими інструментами, веб-скрапінг на основі хмарних технологій вважається найбільш ефективним та надійним рішенням.

Використовуючи інструменти на основі хмарних технологій не потрібно встановлювати жодного програмного забезпечення. Зазвичай налаштування в таких системах дуже прості і водночас мають найбільше варіацій порівняно з двома попередніми типами веб-скраперів. Одразу після налаштування система починає роботу. Дані можна отримати через API у веб-браузері і завантажити у вибраному форматі.

Хмарні технології не обмежують веб-скрапінг певним об’ємом даних завдяки використанню декількох обчислювальних середовищах. Таким чином можна зробити висновок, що використання хмарних технологій для веб-скрапінгу є найбільш доцільним при роботі з великими об’ємами даних та складними веб-сайтами.

У порівнянні з іншими інструментами, які вимагають втручання людини на деяких етапах веб-скрапінгу, інструменти веб-скрапінгу з використанням хмарних технологій можуть полегшити процес підтримки веб-скрапінгу та зробити його повністю автоматизованим і безпроблемним [6].

4. Власне програмне забезпечення.

Потрібно взяти до уваги, що можна створити своє власне програмне забезпечення для веб-скрапінгу використовуючи майже будь яку мову програмування. Таке рішення має свої переваги. Наприклад, користувач може налаштувати програму повністю під себе і створити найкращі умови для збору інформації саме з тих ресурсів, які йому потрібні.

1.4 Веб-скрапери і загрози приватності та безпеці

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.

Веб-скрапінг створює серйозний виклик для компаній, він може зменшувати продажі, знижувати SEO рейтинг, ставити під сумнів правдивість інформації розміщеної на інших веб-сайтах.

Обізнаність в полі веб-скрапінгу дозволяє власникам веб-сайтів не тільки більш ефективно захищати свої дані та ресурси, а й вживати заходів щодо захисту своєї власної інформації.

Веб-скрапінг – це програмна технологія для вилучення інформації з веб-сайтів і організації перетворення неструктурованих даних веб-сайту в базу даних для аналізу або повторного використання вмісту на веб-сайті.

У більшості випадків боти, які складають 46% веб-трафіку, реалізуються окремими людьми для виконання автоматизованого доступу до інтернет ресурсів набагато швидше, ніж люди.

38% компаній, які займаються веб-скрапінгом, роблять це для отримання контенту, а також використовують можливі технології для досліджень, отримання контактних даних, порівняння цін, моніторингу даних погодних сервісів та виявлення змін на сайті.

Найбільше від веб-скрапінгу постраждали наступні галузі: нерухомість, цифрові публікації, електронний документообіг, каталоги та оголошення, авіакомпанії та туристичні компанії.

На даний момент близько 2% онлайн прибутків може бути втрачено через неправильне вкористання інформації в мережі Інтернет.

Це не єдина проблема, оскільки веб-скрапінг здатний викривати приватну інформацію, розміщену в Інтернеті, і може призвести до значних штрафів [7, 17].

1.4.1 Правові особливості веб-скрапінгу

Репутація веб-скрапінгу значно погіршилася протягом останніх кількох років і з достатньо зрозумілих причин:

- Веб-скрапінг все частіше використовується для комерційних цілей, щоб отримати вигоду в боротьбі з конкурентами. Отже, досить часто важливу роль відіграє фінансове питання.
- Веб-скрапери майже у всіх випадках повністю ігнорують закони про авторські права та Правила використання (Terms of Service).
- Веб-скрапери можуть надсилати набагато більше запитів на секунду, ніж людина, що може призвести до непередбачуваного навантаження на веб-сайти.
- Веб-скрапери зазвичай залишаються анонімними і ніяк не ідентифікують себе.
- Веб-скрапери також можуть виконувати заборонені операції на веб-сайтах, наприклад, обходити заходи безпеки, які застосовуються для автоматичного завантаження даних, що є недоступними в загальному доступі.

Дуже багато людей і компаній використовують веб-скрапери у власних цілях. Очевидно, що це викликає занепокоєння в компаніях, чиї веб-сайти стають об’єктами веб-скрапінгу, наприклад, соціальні мережі (наприклад, Facebook, LinkedIn тощо), онлайн-магазини (наприклад, Amazon) та інформаційні ресурси (наприклад, CNN, BBC тощо). Саме через це компанії-гіганти, особливо ті, що мають доступ до персональних даних людей,

наприклад, Facebook, мають окремі Правила використання для автоматизованого збору даних.

З іншого боку, збирання даних в інтернеті вже давно використовується пошуковими системами (наприклад, Google, Bing тощо) для завантаження та індексування даних в мережі інтернет. Ці компанії створювали позитивну репутацію протягом багатьох років. Вони створюють необхідні інструменти, що підвищують рейтинг веб-сайтів, яких вони сканують. Таким чином, веб-скрапінг в Інтернеті, як правило, сприймається більш позитивно, хоча іноді він може використовуватись також зі зловмисними намірами.

Веб-скрапінг не є незаконним. Врешті-решт, на власному сайті можна витягувати даних без жодних обмежень.

Проблема виникає при роботі з іншими веб-сайтами, не отримуючи попереднього письмового дозволу або нехтуючи їхніми Правилами використання (ToS). Таким чином людина, що використовує веб-скрапер ставить себе в уразливе становище.

Інструменти веб-сркапінгу навантажують систему, обмежують пропускну здатність. Також з допомогою веб-скрапера стороння людина витягує дані та вільно використовує їх в своїх цілях. Логічно, що власникам веб-сайтів такий підхід до використання їх ресурсів не завжди подобається, адже це може нанести їм шкоду, в тому числі і матеріальні збитки. Таким чином, в залежності від багатьох факторів, власники веб-сайтів мають всі необхідні причини для звернення до суду.

Є яскраві приклади, що доводять незаконність веб-скрапінгу в деяких випадках:

- Справа LinkedIn проти Doe Defendants. Компанія LinkedIn судиться з людьми, які анонімно автоматизовано збирали дані з веб-сайту.
- Справа Associated Press проти Meltwater U.S. Holdings, Inc. Суд встановив, що портал для розміщення новин Meltwater використовував статті Associated Press незаконно.

- Справа LinkedIn проти Robocog Inc. Компанія Robocog Inc. Була вимушена заплатити 40000 доларів США компанії LinkedIn через несанкціонований автоматизований перегляд веб-сайту.

Є абсолютно законні причини звертатися до судових органів влади в таких випадках:

- Порушення Закону «Про комп'ютерне шахрайство та зловживання» (CFAA).
- Порушення Закону «Про захист авторських прав у цифрову епоху» (DMCA).
- Порушення Правил використання.
- Порушення права власності.

Також абсолютно не має значення з якої країни проводиться такий автоматизований доступ. Важливо те, якій країні належить веб-сайт і якими законами він охороняється. Найбільш захищеними є веб-сайти, що обслуговуються на території Сполучених Штатів Америки (Закон «Про комп'ютерне шахрайство та зловживання» - CFAA, Закон «Про захист авторських прав у цифрову епоху» - DMCA), Канада (Закон «Про захист особистих даних та електронних документів» - PIPEDA) та Європейського союзу (Загальний регламент про захист даних – GDPR).

Загальний регламент захисту даних, або GDPR, як він більш відомий, застосовується тільки до персональних даних. Тобто до даних, які тим чи іншим чином можуть ідентифікувати особу. Прикладами особистих даних є:

- ім'я;
- фізична адреса;
- адреса електронної пошти;
- номер телефону;
- деталі кредитної картки;
- деталі банківського рахунку;
- дата народження;

- інформація про зайнятість;
- номер соціального страхування;
- медична інформація;
- фото та відео чи аудіо записи.

Крім того, виходячи з вищенаведеного судового позову з боку LinkedIn, очевидно, що такі випадки, безсумнівно, можуть стати досить складними і об’ємними.

Можна виділити наступні ситуації, коли веб-скрапінг може стати незаконним:

- Веб-сайт, включаючи його сторінки, дизайн, макет і базу даних – можуть бути захищені авторським правом, оскільки він розглядається як творча робота. Вилучення з нього даних і простий факт копіювання веб-сторінки за допомогою веб-скрапера може вважатися порушенням авторських прав.
- Використання статтей та будь якого іншого авторського матеріалу у власних цілях
- Отримання персональних даних людей, для створення клієнтських баз даних та подальшого їх використання
- Порушення Правил використання, навіть якщо файл robots.txt не забороняє веб-скрапінг на веб-сайті [8].

1.4.2 Законність веб-скрапінгу

Хоча численні інструменти та технології були доступні для організації автоматизованого доступу до інтернет ресурсів за допомогою веб-скрапінгу, законність веб-скрапінгу все ще залишається «сірою зоною» в правовому полі. Не існує законодавчих актів, які б безпосередньо стосувалися веб-скрапінгу. Наразі веб-скрапінг керується набором пов’язаних, фундаментальних правових теорій і законів. Для прикладу наведемо деякі з них:

- Порушення авторських прав;
- Порушення договору;
- Закон «Про комп’ютерні шахрайства та зловживання» (CFAA);
- Порушення прав власності.

Нижче наведені деякі конкретні подробиці того, як ці фундаментальні правові теорії застосовуються до веб-скрапінгу:

1. Правила використання.

У правовому полі часто говориться, що власник веб-сайту може ефективно запобігати програмному доступу до веб-сайту, явно забороняючи це в політиці «Правил використання», розміщеної на веб-сайті. Недотримання цих умов може призвести до порушення контракту зі сторони користувача веб-сайту. Щоб притягнути когось до відповідальності за порушення «умов використання», користувачеві веб-сайту необхідно укласти явну угоду з власником веб-сайту про дотримання політики «Загальних положень та умов» (наприклад, погодитись з цими правилами на сайті). Проста заборона веб-сканування і веб-скрапінгу на сайті не може перешкоджати комусь автоматизовано отримувати інформацію з юридичної точки зору.

2. Матеріал, захищений авторським правом.

Веб-скрапінг та повторна публікація матеріалів, якими володіє власник веб-сайту та які є явно захищені авторським правом власника веб-сайту, може призвести до відкриття справи про порушення авторських прав.

3. Веб-скрапінг з визначеною метою.

Будь-яке незаконне або шахрайське використання даних, отриманих за допомогою веб-скрапінгу, заборонено законом. Наприклад, особа, яка отримує доступ до даних з Інтернету, які є конфіденційними та захищеними, може бути притягнута до відповідальності за законом про комп’ютерні шахрайства та зловживання, якщо збиток перевищує 5000 доларів США. В Інтернеті це досить поширена практика, коли людина свідомо отримує доступ до «преміум-контенту», а потім перепродає його, використовує у своїх цілях його або

продовжує отримувати доступ до контенту через несанкціонований канал навіть після отримання попереджувального листа від власника веб-сайту.

4. Завдання шкоди ресурсам та веб-сайту.

Якщо веб-скрапінг перевантажує або пошкоджує веб-сайт або сервер, то особа, відповідальна за нанесені збитки, може бути притягнута до відповідальності за законом про порушення прав власності. Тим не менш, збиток повинен бути матеріальним і таким, що його можна буде легко довести в суді, для того щоб власник веб-сервера мав право на фінансову компенсацію.

5. Індивідуальна конфіденційність

Проекти, що використовують дані, зібрані з веб-сайтів, можуть ненавмисно порушити конфіденційність приватних осіб. Навіть якщо не порушено індивідуальну конфіденційність, проблема полягає в тому, що користувачі веб-сайту можуть не погодитися на використання їх даних третіми особами. Таким чином, використання цих даних без згоди є порушенням прав. Ці порушення конфіденційності та прав можуть призвести до серйозних наслідків для власника веб-сайту, враховуючи останні скандали в сфері конфіденційності та інформації про користувачів (справа Facebook та Cambridge Analytica).

6. Комерційна таємниця.

Так само як і люди, компанії також мають право зберігати певні аспекти своєї діяльності в конфіденційності. З допомогою веб-скрапінгу можна ненавмисно розкрити комерційні таємниці або просто конфіденційну інформацію про організацію. Це може пошкодити репутацію компанії і призвести до матеріальних та фінансових втрат [9].

Висновки до розділу 1

В даному підрозділі були розглянуті такі поняття як веб-скрапер та веб-скрапінг. Були розглянуті види програмного забезпечення, що реалізують

автоматизований доступ до веб-ресурсів. Також були проаналізовані сфери використання веб-скраперів та загрози, які несе веб-скрапінг.

Було виділено наступні ситуації, коли веб-скрапінг може стати незаконним:

- Веб-сайт, включаючи його сторінки, дизайн, макет і базу даних – можуть бути захищені авторським правом, оскільки він розглядається як творча робота. Вилучення з нього даних і простий факт копіювання веб-сторінки за допомогою веб-скрапера може вважатися порушенням авторських прав.
- Використання статей та будь якого іншого авторського матеріалу у власних цілях
- Отримання персональних даних людей, для створення клієнтських баз даних та подальшого їх використання
- Порушення Правил використання, навіть якщо файл robots.txt не забороняє веб-скрапінг на веб-сайті.

2 МЕТОДИ ВИЯВЛЕННЯ ТА БЛОКУВАННЯ ВЕБ-СКРАПЕРІВ

Атаки з використанням веб-скраперів можуть знизити доходи і прибутки компанії шляхом крадіжки відвідувачів сайтів і копіювання цінного контенту. Веб-скрапінг може також підірвати конкурентоспроможність бізнесу, тому що конкуренти можуть використовувати інформацію про ціни, щоб відкоригувати ціни на своїй стороні. Таким чином, атаки з використанням веб-скраперів можуть призвести до чималих втрат для бізнесу.

Існує чимало способів виявлення, обмеження та блокування веб-скраперів на сайті: вжиття юридичних заходів проти веб-скраперів, вжиття технічних засобів, які б обмежували можливості веб-скраперів та блокування веб-скраперів різними способами.

Оскільки не існує чітких законів, які б перешкоджали веб-скрапінгу, використання правових можливостей для зупинки веб-скраперів може бути складним, дорогим і вимагати дуже багато часу, хоча в деяких випадках такі зусилля цього варті. Власники веб-сайтів, що піддались атакам веб-скраперів повинні чітко продемонструвати, що веб-скрапери порушили політики та Правила використання веб-сайту.

В якості альтернативи компанії можуть запобігати веб-скрапінгу, створюючи перепони та бар’єри для обмеження та блокування автоматизованого доступу до інтернет ресурсів.

Існують такі поширені способи як CAPTCHA та обфускація коду, які можуть заважати ваб скраперам отримувати дані використовуючи скрипти. Тим не менш, для того щоб обійти такі заходи безпеки можуть бути розроблені більш складні та вдосконалені інструменти для веб-скрапінгу.

Незважаючи на те, що прості засоби веб-скрапінгу відносно легко виявити, вдосконалене програмне забезпечення для автоматизованого збору даних в мережі Інтернет може вимагати використання таких методів виявлення в комбінації з іншими.

2.1 Виявлення веб-скраперів

Існує достатньо багато методів виявлення веб-скраперів, та не всі є ефективними. В ході виконання роботи було розглянуто та виокремлено найбільш ефективні з них. Нижче наведено деякі приклади:

1. Використання файлів cookie та мови програмування JavaScript.

Ці засоби використовують для того, щоб переконатися, що користувач відвідує сайт через звичайний веб-браузер. Більшість простих інструментів для веб-скрапінгу не можуть обробляти складний JavaScript код або зберігати файли cookie. Щоб переконатися, що запити до серверу виконуються через веб-браузер можна помістити на сторінках будь-які обчислення мовою JavaScript і визначити, чи код правильно скомпільовано та обчислено.

2. Використання CAPTCHA.

CAPTCHA використовують для того щоб переконатися, що користувач є людиною, а не комп'ютером. CAPTCHA може значно ускладнити реалізацію атак з використанням веб-скраперів. Проте, боти все частіше знаходять способи обійти CAPTCHAs. До 60 відсотків ботів можуть пройти через CAPTCHA, згідно з нещодавніми дослідженнями безпеки. Проте CAPTCHA все ще є ефективним захистом від багатьох видів веб-скраперів.

3. Заманювання веб-скраперів.

Використання інформації або зображень для заманювання веб-скраперів на сайти. Якщо компанія підозрює, що її власні дані використовуються кимось іншим для своїх потреб і розміщуються на інших ресурсах, то вони можуть використати такий фальшивий контент як доказ в суді.

Працюють спеціальні моніторингові компанії, що надають послуги виявлення плагіату. Оскільки деякі веб-скрапери автоматично сканують весь веб-сайт, створення прихованого тексту чи прихованих посилань на спеціально створені сторінки з неправдивим вмістом може допомогти виявити порушників.

4. Використання пасток.

Пастки – потрібний і корисний інструмент для виявлення і запобігання збору інформації веб-скраперами та веб-роботами. При потраплянні такої системи на пастку можна записувати IP-адреси та обмежувати і блокувати для них доступ до інтернет ресурсів.

Хоча й комерційні системи захисту не публікують технічні деталі про свої методи виявлення веб-скраперів, можна припустити, що такі методи цілком можливо можуть успішно використовуватись [10].

Цей метод буде детально розглянуто далі.

2.2 Блокування веб-скраперів

Існує достатньо багато методів блокування веб-скраперів, та не всі є ефективними. В ході виконання роботи було розглянуто та виокремлено найбільш ефективні з них. Нижче наведено деякі приклади:

1. Юридичний захист.

Найбільш простий метод запобігти атакам веб-скраперів – зайняти чітку юридичну позицію, коли на сайті чітко визначено, що веб-скрапінг заборонений. Наприклад, в Правилах використання сайту Medium вказано, що сканування веб-сайту дозволено, якщо воно виконується згідно з файлом robots.txt, але веб-скрапінг заборонений.

2. Обмеження кількості запитів.

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 все одно будуть виявляти себе, звертаючись до веб-сторінки після відповідного проміжку часу. Веб-скрапери також частіше створюють набагато більшу кількість одночасних з’єднань і отримують доступ до більшої кількості сторінок, ніж більшість

користувачів. Деякі веб-скрапери отримують доступ до HTML-файлів і ігнорують зображення та інші файли. Таку поведінку теж можна відслідковувати.

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. Також, оскільки більшість інструментів веб-скрапінгу не можуть інтерпретувати JavaScript або CSS, веб-розробники можуть створювати сторінки таким чином, щоб текст компілювався за допомогою скриптів або стилів, щоб унеможливити викрадення контенту з веб-сайту [11].

4. Зміна HTML-тегів та DOM-структури веб-сайту.

Часто використовують зміну HTML-тегів та DOM-структури веб-сайту для запобігання повторних атак веб-скраперів. Веб-скрапери запрограмовані на розбір сайтів і отримання необхідних даних. Зміна HTML-тегів і структури веб-сайту, наприклад, додавання пробілів і коментарів, зміна імені тегів, класів, ідентифікаторів та URL-адреси може запобігти повторним атакам веб-скраперів. Для проведення повторної атаки потрібно ще раз повністю проаналізувати структуру веб-сайту.

5. Блокування IP-адрес засобами мови програмування PHP.

За допомогою мови програмування PHP можна блокувати доступ до веб-сайту для користувачів та систем з заранне підготовленого списку IP-адрес. Можна змінювати заголовки запитів чи перенаправляти всі запити на створену для цього сторінку.

Саме цей метод використовувався для блокування виявлених системою виявлення веб-скраперів з використанням пасток IP-адрес, архітектура якої буде описана в наступному розділі.

6. Запобігання атакам відмов у обслуговуванні (DoS)

Навіть якщо було згадано про заборону використання веб-скраперів на сайті в Правилах використання, потенційний зловмисник не зверне на це

увагу, що при великих навантаженнях може спричинити DoS-атаку. Необхідно уникнути таких ситуацій. Можна визначити та заблокувати потенційні IP-адреси, фільтруючи їх через брандмауер [12].

7. Використання CSRF токенів.

Використовуючи токени, що перевіряють правильність запитів можна значно уповільнити чи унеможливити автоматизований доступ веб-скраперів до веб-сторінок інтернет-ресурсу. Такий токен може використовуватись як ідентифікатор сеансу чи приховане поле форми. Для обходу такого методу захисту потрібно завантажити та проаналізувати розмітку на сторінці, знайти правильний токен і тільки тоді додати його до запиту. Цей процес вимагає значних зусиль, навичок у програмуванні і доступ до більш професійних інструментів.

8. Використання .htaccess файлів для запобігання атакам веб-скраперів

.htaccess – це конфігураційним файл для веб-сервера Apache, і його можна налаштувати для запобігти доступу веб-скраперів до сайту. Першим кроком є виявлення веб-скраперів. Після їх ідентифікації можна скористатися багатьма методами щоб заблокувати доступ для веб-скраперів до веб-сайту (див рис 2.1).

```
1  Order Allow,Deny
2  Allow from all
3  Deny from [IP-адреса чи підмережа]
```

Рисунок 2.1 – Приклад блокування для веб-серверу Apache

За замовчуванням файл .htaccess не ввімкнено на Apache і його використання потрібно дозволити, після чого Apache буде інтерпретувати файли .htaccess, розміщені на сервері.

Файли .htaccess можна створювати лише для Apache. Для веб-серверів Nginx можна використовувати ngx_http_access_module для вибіркового дозволу або заборони запитів з певної IP-адреси (див. рис. 2.2).

```
1  deny [IP-адреса];
2  deny subnet;
3  allow [IP-адреса];
4  allow subnet;
5  # block all ips
6  deny    all;
7  # allow all ips
8  allow    all;
```

Рисунок 2.2 – Приклад блокування для веб-серверу nginx

Аналогічно, для серверів IIS можна обмежити доступ IP-адрес додавши нову роль в Менеджері серверів (див. рис. 2.3).

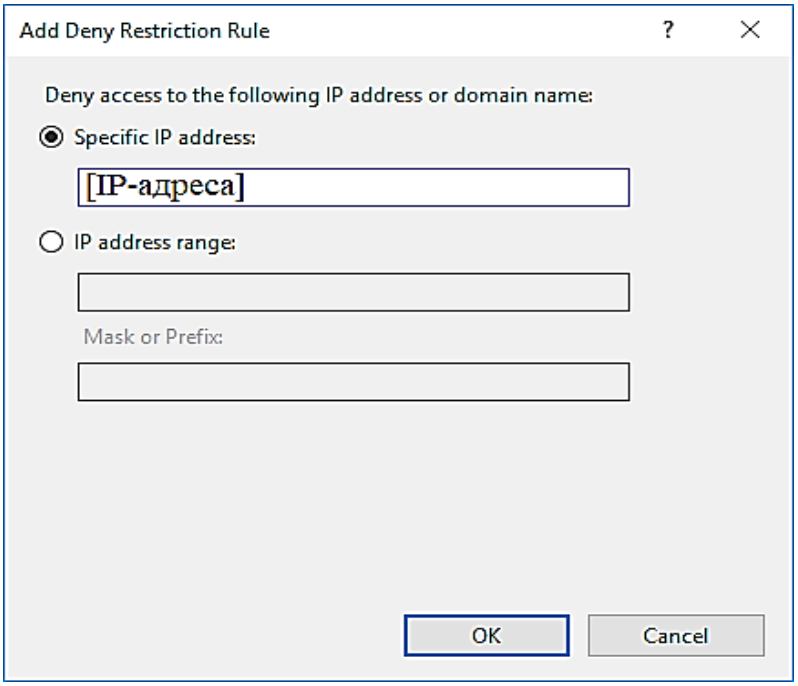

Рисунок 2.3 – Приклад блокування для веб-серверу IIS

9. Заборона використання посилань на ресурси з інших веб-сайтів.

Використання посилань на ресурси з інших веб-сайтів називають (hotlinking) є досить поширеною практикою в мережі інтернет. Та якщо відомий факт про використання контенту з веб-сайту незаконно – такий підхід можна заблокувати.

Коли вміст веб-сайту стягується, вбудовані посилання на зображення та інші файли копіюються безпосередньо на сайт зловмисника. Коли на сайті зловмисника відображається той самий вміст, такий ресурс (зображення або інший файл) безпосередньо посилається на батьківський веб-сайт. Цей метод відображення ресурсу, розміщеного на сервері на іншому веб-сайті, називається гарячим посиланням.

Блокування можливе для більшості поширених веб-серверів (Apache, nginx, IIS).

Для веб-серверів Apache код додається до глобального файлу конфігурацій веб-серверу – httpd.conf або в локальний файл конфігурацій – .htaccess (див. рис. 2.4).

```
1  RewriteEngine on
2  RewriteCond %{HTTP_REFERER} !^$
3  RewriteCond %{HTTP_REFERER} !^[ваш веб-сайт]/.*$ [NC]
4  RewriteRule .*\.(gif|jpe?g|png)$ [ваш веб-сайт]/hotlinking.jpg [R,NC,L]
```

Рисунок 2.4 – Приклад реалізації заборони гарячих посилань для веб-серверів Apache

Для веб-серверів nginx код додається до файлу конфігурацій веб-серверу nginx.conf (див. рис. 2.5).

```
1  location ~ .(gif|png|jpe?g)$ {
2      valid_referers none blocked [ваш веб-сайт] *.[ваш веб-сайт];
3      if ($invalid_referer) {
4         return   403;
5      }
6  }
```

Рисунок 2.5 – Приклад реалізації заборони гарячих посилань для веб-серверів nginx

Для веб-серверів IIS код додається до файлу конфігурацій web.config (див. рис. 2.6).

```
<rules>
  <rule name="Prevent Image Hotlinking">
    <match url=".*\.(gif|jpg|png)$"/>
    <conditions>
      <add input="{HTTP_REFERER}" pattern="^$" negate="true"/>
      <add input="{HTTP_REFERER}" pattern="^[ваш веб-сайт].*$" negate="true"/>
    </conditions>
  </rule>
</rules>
```

Рисунок 2.6 – Приклад реалізації заборони гарячих посилань для веб-серверів IIS

Таким чином, веб-сервер не буде обслуговувати будь-які ресурси з вашого веб-сайту на інших інтернет-ресурсах. Але не рекомендується блокувати весь контент з веб-сайту, так як гарячі посилання широко використовуються методами пошукової оптимізації веб-сайтів (SEO). Найкращим рішенням є блокування основних та найбільш цінних ресурсів.

10. Повідомлення про зловмисника пошуковим системам і провайдерам.

За допомогою можна звернутись до пошукових систем, щоб вони видалили контент, який був викрадений з сайту. Також можна звернутись до інтернет провайдерів, щоб вони повністю заблокували доступ до веб-сайту для потенційних веб-скраперів [13].

Оскільки використання якогось одного методу виявлення чи блокування веб-скраперів може бути не зовсім ефективним, їх можна використовувати в комбінації з іншими для точнішого визначення атаки. Деякі звичайні користувачі відключають файли cookie та JavaScript. Проте систематичні запити до веб-серверу виконані з певним інтервалом все одно будуть вказувати на автоматизований характер таких запитів, що свідчить про атаку на веб-сайт

з використанням веб-скраперів. Якщо веб-сайти будуть розроблятись з використанням наведених вище методів, вони можуть значно зменшити кількість таких атак, а в деяких випадках повністю їх усунути.

2.3 Виявлення веб-скраперів з використанням пасток

Веб-скрапери це автоматизовані програми, які систематично переглядають веб-сторінки і збирають інформацію. Хоча такі роботи є важливими інструментами для індексування контенту в інтернеті, вони можуть бути шкідливими через фішинг, спам або виконання цільових атак. У цій роботі буде розглянуто підхід до виявлення веб-скраперів, який використовує пастки у вигляді прихованих посилань (ресурсів) на веб-сторінках.

Звичайно існують веб-роботи, які можуть обходи деякі методи захисту на основі пасток. Удосконалені веб-роботи можуть використовувати cookie, виконувати код JavaScript, емулювати клавіатуру і комп’ютерну мишку, аналізувати код та обходити сайт лише по попередньо визначених після аналізу директоріях [14].

На даний момент є дуже мало досліджень, присвячених ефективності простих технологій пасток для виявлення як потрібних так і шкідливих веб-роботів. У розробників веб-сайтів є всі права приховувати контент всередині веб-сторінок. Приховані поля можна використовувати в веб-формах для зберігання значень визначених за замовчуванням, які можна динамічно змінювати в залежності від дій користувача.

2.3.1 Види пасток

Веб-скрапери це автоматизовані програми, які систематично переглядають веб-сторінки і збирають інформацію. Хоча такі роботи є важливими інструментами для індексування контенту в інтернеті, вони можуть бути шкідливими через фішинг, спам або виконання цільових атак. У цій роботі буде розглянуто підхід до виявлення веб-скраперів, який використовує пастки у вигляді прихованих посилань (ресурсів) на веб-сторінках.

Пастка – це механізм безпеки розроблений для полегшення виявлення кого-небудь чи чого-небудь, що виконує автоматизований доступ до ресурсів. Пастка – це 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 можна виділити «пастку 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). В результаті веб-адміністратори, що використовують «пастки для павуків» часто додають свої сторінки-пастки в список виключень файлу robots.txt.

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).

Більшість веб-скраперів не передбачають перевірку форматування сторінки, вони просто стягують код сторінки, тобто вони не зможуть перевірити чи є посилання чи ресурс прихованим перед тим як робити запит.

Як було сказано раніше, існують веб-роботи, які можуть обходи деякі методи захисту на основі пасток. Удосконалені веб-роботи можуть використовувати cookie, виконувати код JavaScript, емулювати клавіатуру і комп'ютерну мишку, аналізувати код та обходити сайт лише по попередньо визначених після аналізу директоріях [15].

Найбільш поширеними є веб-скрапери, що обходять веб-сайт по конкретним директоріям. Саме для такого вдосконалення веб-скраперів в ході проведення дослідження був запропонований удосконалений метод виявлення веб-скраперів з використанням пасток. На основі цього методу була розроблена система виявлення веб-скраперів з використанням пасток, архітектура якої буде розглянута в наступному розділі.

2.3.2 Способи представлення пасток

Веб-розробники використовують метод приховування контенту на інтернет сторінках передусім для залучення на свій сайт потрібних веб-роботів (наприклад Google роботів, сканерів інших пошукових систем). Практика приховування контенту всередині веб-сторінок використовується веб-розробниками для підвищення пошукової оптимізації сторінок веб-сайту (SEO) чи підвищення рейтингу сторінки в пошукових системах.

Розробники веб-сайтів приховують контент, який можуть переглядати лише веб-сканери на веб-сторінках щоб підвищити рейтинг сторінок в результатах пошуку (заповнення ключовими словами). Пошукові системи

адаптуються до цієї практики, змінюючи алгоритми сканування щоб виявити такий тип заповнення ключових слів.

Ці методи приховування включають в себе такі способи як встановлення нульового розміру блоку, створення відступів елементів сторінок таким чином, щоб вони не з’являлись на екрані, приховування тексту з ключовими словами, зробивши його таким самим кольором як і фон сторінки.

Можна більш детально розглянути найбільш дієві методи приховування елементів на веб-сторінках (табл. 2.1).

Таблиця 2.1 – Механізми приховування контенту

Правила мови CSS	**Відображення на веб-сторінці**

.honeypot {
display: none;
}
	Елемент повністю видаляється з поточної структури сторінки. Простір, який він займав, переходить до наступного блоку.

.honeypot {
height: 0;
width: 0;
overflow: hidden;
}
	Елемент згортається, весь зміст приховується.

.honeypot {
position: absolute;
top: -9999px;
left: -9999px;
}
	Елемент розташовується за межами екрану, таким чином він стає невидимим для звичайного користувача.

.honeypot {
text-indent: -999em;
}
	Елемент зміщується за межі екрану і приховується від звичайного користувача. Але на деяких екранах і в деяких веб-браузерах такий метод відпрацьовує некоректно.

Продовження таблиці 2.1

Правила мови CSS	**Відображення на веб-сторінці**

.honeypot {
z-index: -1;
}
	Елемент розміщується за елементами з більшим індексом. Всі елементи мають z-index: 0 за замовчуванням.

.honeypot {
position: absolute;
overflow: hidden;
clip: rect(0 0 0 0);
height: 1px;
width: 1px;
margin: 1px;
padding: 0;
border: 0;
}
	Елемент обрізається і згортається. Блок не впливає на розмітку.

2.4 Удосконалений метод 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 дослідження 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 директорії 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 2.7.

.htaccess

```
1  Redirect 301 /1/honeypot.html   http://diploma.local/honeypot.html
2  Redirect 301 /2/honeypot.html   http://diploma.local/honeypot.html
3  Redirect 301 /2/21/honeypot.html    http://diploma.local/honeypot.html
4  Redirect 301 /2/22/honeypot.html    http://diploma.local/honeypot.html
5  Redirect 301 /3/honeypot.html   http://diploma.local/honeypot.html
6  Redirect 301 /3/31/honeypot.html    http://diploma.local/honeypot.html
7  Redirect 301 /3/31/311/honeypot.html    http://diploma.local/honeypot.html
8  Redirect 301 /3/31/311/3111/honeypot.html   http://diploma.local/honeypot.html
9
10
```

Рисунок 2.7 – 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 2.8. Для прикладу взято одне з запропонованих на рисунку 2.7 перенаправлень.

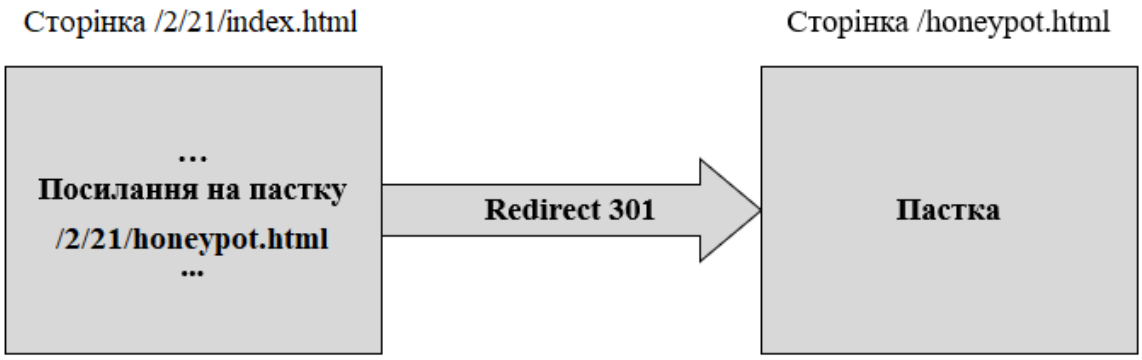

Рисунок 2.8 – Схема роботи запропонованого методу представлення пасток

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.

Ефективність методу полягає у 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 веб-сайті, використовуючи звичайний механізм додавання пасток, неможливо [16].

Висновки до розділу 2

В даному розділі було розглянуто найбільш поширені методи виявлення та блокування веб-скраперів. Також були представлені різні види пасток та методи їх представлення на веб-сторінці.

В заключній частині розділу було представлено вдосконалений метод виявлення веб-скраперів з використанням пасток. Даний метод забезпечує стовідсоткове покриття всіх сторінок веб-сайту пастками і при цьому не потребує створення чи використання додаткових сторінок. Ефективність методу полягає в виявленні тих веб-скраперів, що сканують веб-сайт лише по певних директоріях. Таким чином, якщо веб-скрапер опрацьовує лише ті директорії сайту, які йому потрібні, він в будь-якому разі потрапить в одну з пасток.

Використання даного методу як основної складової системи виявлення веб-скраперів з використанням пасток буде представлено в наступному підрозділі.

3 АРХІТЕКТУРА СИСТЕМИ ВИЯВЛЕННЯ ВЕБ-СКРАПЕРІВ З ВИКОРИСТАННЯМ ПАСТОК

Загроза несанкціонованого збору інформації з веб-ресурсів в даний час є дуже актуальною. З кожним роком кількість трафіку, класифікованого як веб-скрапінг, згідно з дослідженнями, значно збільшується. Понад 22% всіх запитів класифікуються як автоматизовані, вони генерують в середньому 27% трафіку. Причому, зростання активності веб-роботів спостерігається вже декілька років поспіль [17].

Також поліпшуються методи і механізми захисту веб-роботів від виявлення. Для цього використовуються ботнети і зламані комп'ютери користувачів. Також, впливає широке розповсюдження легких у використанні плагінів автоматизації збору інформації для веб-браузерів.

Найбільш популярними цілями є сайти для продажу квитків, електронної комерції, сайти для розміщення оголошень в сфері подорожей і нерухомості, а також соціальні мережі.

Саме через це створення системи виявлення веб-скраперів є досить актуальним питанням в наш час. Так само як вдосконалюються веб-скрапери мають вдосконалюватися і методи їх виявлення.

Запропонований в попередньому розділі вдосконалений метод виявлення веб-скраперів з використанням пасток став основою для створення даної системи виявлення. Створена система може не обмежуватись лише даним методом виявлення, вона може комбінувати в собі декілька методів виявлення та блокування веб-роботів.

В даному розділі буде розглянута система, в основі якої лежить саме цей запропонований вдосконалений метод виявлення веб-скраперів, що дозволяє виявляти веб-роботів і з використанням додаткових можливостей системи також і автоматично блокувати їх одразу після виявлення.

Розглянемо процеси, що передбачені системою виявлення веб-скраперів з використанням пасток (див. рис. 3.1).

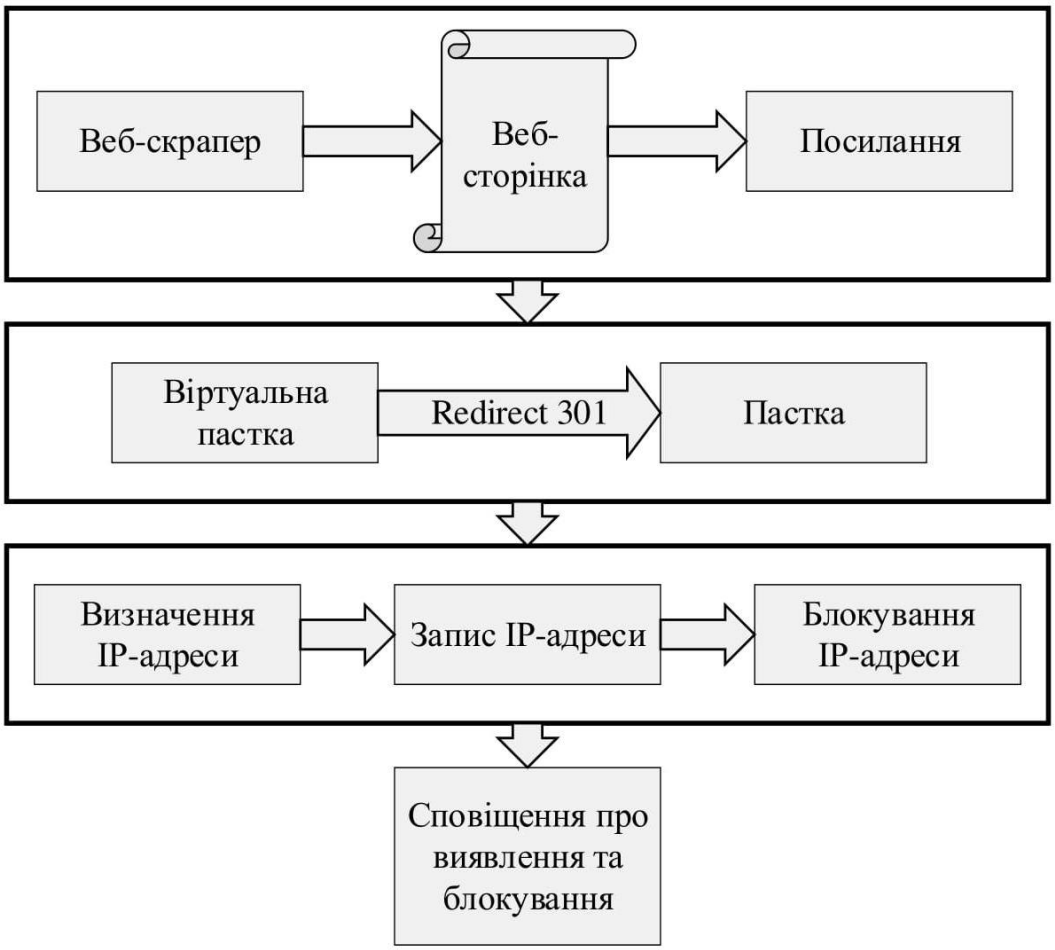

Рисунок 3.1 – Схема процесів системи виявлення веб-скраперів з використанням пасток

Спочатку веб-скрапер потрапляє на одну з сторінок веб-сайту. Далі програма збирає всі посилання з веб-сторінки і переходить по кожному з них. На наступному етапі веб-скрапер потрапляє на одну з пасток (віртуальну, якщо знаходиться в якійсь з директорій, чи одразу на фізичну сторінку-пастку, якщо програма знаходиться в кореневому каталозі).

При потраплянні на віртуальну пастку, веб-скрапер перенаправляється на фізичну сторінку /honeypot.html. При переході на фізичну сторінку

виконується функція getUserIP(), яка визначає IP-адресу, з якої виконується звернення до серверу.

Наступний крок – це запис отриманої IP-адреси в файл ip_list.txt, в якому зберігаються всі IP-адреси потенційних веб-скраперів. Далі IP-адреса додається в масив заблокованих.

І останній процес, що передбачений системою виявлення веб-скраперів з використанням пасток – це надсилання сповіщення власнику веб-сайту про виявлення та блокування веб-скрапера.

В наступних підрозділах цього розділу кожен з процесів буде розглянуто як складова окремої підсистеми.

3.1 Складові системи виявлення веб-скраперів з використанням пасток

Запропонована система виявлення веб-скраперів складається з п'яти підсистем (див. рис. 3.2), кожна з яких виконує передбачені архітектурою функції.

Короткий опис кожної з них:

1. Підсистема розповсюдження пасток.

Основна функція цієї підсистеми – додавання прихованих блоків з пасткою на кожну сторінку веб-сайту. Цей функціонал запрограмований в файлі trap.php, що автоматично підключається до кожної сторінки.

Рисунок 3.2 – Складові системи виявлення веб-скраперів

2. Підсистема виявлення присутності веб-скраперів на веб-сайті.

Основна функція – визначення IP-адреси користувача чи системи, що потрапили на сторінку-пастку та запис отриманої IP-адреси у відповідний файл (ip_list.txt). Для визначення IP-адреси, до сторінки-пастки (honeypot.html) підключається файл getIP.php і використовується функція *getUserIP()*.

3. Підсистема автоматичного блокування веб-скраперів.

Основна функція – блокувати доступ до веб-сторінок користувачам та системам, чиї IP-адреси потрапили до файлу ip_list.txt. Відповідний функціонал запрограмований у файлі deny.php.

4. Підсистема моніторингу структури сайту.

Ця підсистема реагує на зміни у структурі сайту та інформує про це власника веб-сайту. Це необхідно для створення перенаправлень з нових віртуальних пасток.

5. Підсистема сповіщення про виявлення веб-скрапера.

Основна функція – інформування власника веб-сайту про виявлення та блокування нового веб-скрапера. Функціонал реалізовано у файлі sendNotification.php.

Також необхідно розглянути зв’язки підсистем системи виявлення між собою (див. рис. 3.3).

Рисунок 3.3 – Зв’язки підсистем системи виявлення веб-скраперів

Як бачимо зі схеми, підсистема моніторингу структури сайту зв’язана з підсистемою розповсюдження пасток саме в такому порядку. Адже засобами підсистеми моніторингу встановлюються перенаправлення з віртуальних сторінок-пасток на фізичну. В той час як підсистема розповсюдження пасток використовує посилання на віртуальні пастки.

Іншу групу підсистем складають підсистема виявлення, підсистема автоматичного блокування та підсистема сповіщення. Після потрапляння веб-скрапера на сторінку-пастку, система визначає IP-адресу веб-скрапера, блокує її та надсилає сповіщення адміністратору веб-сайту про виявлення та блокування веб-скрапера.

3.2 Підготовка системи до експлуатації

Підготовка системи виявлення веб-скраперів з використанням пасток до експлуатації складається з декількох етапів:

- Налаштування .htaccess файлу;
- Створення необхідних для функціонування системи HTML сторінок та PHP файлів;
- Налаштування файлу robots.txt;
- Підключення бібліотеки simple_html_dom.php для роботи з елементами веб-сторінки.

Першим кроком є налаштування .htaccess файлу. Для початку потрібно дозволити інтерпретацію PHP коду на HTML-сторінках. Також потрібно підключити файл loader.php, який повинен автоматично додаватись до кожної веб-сторінки. Приклад сконфігурованого на даному етапі .htaccess файлу представлено на рисунку 3.4.

Рисунок 3.4 – Приклад сконфігурованого .htaccess файлу

Файл loader.php збирає PHP-файли, що необхідні на кожній сторінці. Він виглядає наступним чином:

```
<?php
   require 'trap.php';
   require 'deny.php';
```

Також необхідно створити файли honeypot.html та denied.html, які використовуються підсистемами виявлення та автоматичного блокування.

Наступним етапом підготовки системи до використання є налаштування файлу robots.txt. Це необхідно для того щоб заблокувати доступ до пастки для «правильних веб-скраперів», тобто тих, хто не ігнорує вміст файлу robots.txt (наприклад, Google роботи). Таким чином вони не будуть виявлені засобами системи виявлення веб-скраперів і зможуть індексувати контент веб-сайту для пошукової системи. Вміст файлу виглядає наступним чином:

```
User-agent: *
Disallow: /honeypot.html
```

І останній етап – підключення бібліотеки simple_html_dom.php для роботи з елементами веб-сторінки. Дана бібліотека потрібна лише для підсистеми розповсюдження пасток, тому підключається лише в файлі trap.php.

Більш детально всі підсистеми будуть описані в наступний підрозділах.

3.3 Опис підсистеми розповсюдження пасток

Підсистема розповсюдження пасток працює наступним чином. В файлі trap.php прописується блок з прихованою пасткою. Цей файл підключений до файлу loader.php, який в свою чергу вказаний в файлі .htaccess як файл, який потрібно підключати на кожну сторінку веб-сайту при її завантаженні. Таким чином даний блок з прихованою пасткою буде вставлятись на кожну сторінку веб-сайту.

Важливою деталлю є вибір методу приховування пасток на веб-сторінці. В попередніх розділах були представлені найбільш поширені методи приховування елементів з використанням мови CSS (див. табл. 2.1). Для досягнення найкращого результату та створення найбільш ефективної системи виявлення веб-скраперів було проведено експеримент з використанням програмного забезпечення для веб-скрапінгу.

Було створено тестову веб-сторінку, в якій було шість прихованих блоків (з використанням кожного з методів). В ході експерименту було запущено програми для веб-скрапінгу та перевірено, чи змогли вони отримати інформацію, що містилась в прихованих блоках.

Для проведення експерименту було вибрано три досить розповсюджених веб-скрапера: FMiner, Octoparse та Visual Scraper. Результати експерименту представлені в таблиці 3.1.

Таблиця 3.1 – Ефективність методів приховування пасток на веб-сторінках

Спосіб приховування пастки	**Потрапляння веб-скрапера на таку пастку**		
	FMiner	Octoparse	Visual Scraper

.honeypot {
display: none;
}
	+	-	-

.honeypot {
height: 0;
width: 0;
overflow: hidden;
}
	+	+	+

Продовження таблиці 3.1

Спосіб приховування пастки	**Потрапляння веб-скрапера на таку пастку**		
	FMiner	Octoparse	Visual Scraper

.honeypot {
position: absolute;
top: -9999px;
left: -9999px;
}
	+	+	+

.honeypot {
position: absolute;
overflow: hidden;
clip: rect(0 0 0 0);
height: 1px;
width: 1px;
margin: 1px;
padding: 0;
border: 0;
}
	+	+	+

.honeypot {
text-indent: -999em;
}
	+	+	+

.honeypot {
z-index: -1;
}
	+	+	-

Як бачимо з результатів експерименту, найменш ефективними методами приховування пасток є видалення елемента з DOM сторінки – властивість display: none (пастка не виявлена програмами для веб-скрапінгу у двох з трьох

випадків) та приховування елементу виставляючи його за іншими – властивість z-index: -1 (пастка не виявлена програмами для веб-скрапінгу в одному з трьох випадків).

Такі методи приховують елементи не лише від користувача, а й від більш удосконаленого програмного забезпечення для веб-скрапінгу.

Також важливим моментом є чи будуть ігноруватись такі елементи програмами для людей з вадами зору. Було зроблено припущення, що явні методі приховування елементів на веб-сторінці, такі як видалення елементу з DOM сторінки, приховування його більш ймовірно будуть ігноруватись програмами для людей з вадами зору. Натомість методи, що виводять елементи за межі екрану зможуть бути прочитані такими програмами.

Для підтвердження цього припущення в ході виконання роботи було проведено експеримент з використанням програми для читання екрану Apple Voice Over. Результати представлено в таблиці 3.2.

Таблиця 3.2 – Поведінка програмного забезпечення для людей з вадами зору Apple Voice Over по відношенню до прихованих блоків

Спосіб приховування пастки	**Потрапляння програми Apple Voice Over на пастку**

.honeypot {
display: none;
}
	Блок з пасткою ігнорується

.honeypot {
height: 0;
width: 0;
overflow: hidden;
}
	Блок з пасткою ігнорується

Продовження таблиці 3.2

Спосіб приховування пастки	**Потрапляння програми Apple Voice Over на пастку**

.honeypot {
position: absolute;
top: -9999px;
left: -9999px;
}
	Програма бачить прихований блок

.honeypot {
position: absolute;
overflow: hidden;
clip: rect(0 0 0 0);
height: 1px;
width: 1px;
margin: 1px;
padding: 0;
border: 0;
}
	Програма бачить прихований блок

.honeypot {
text-indent: -999em;
}
	Програма бачить прихований блок

.honeypot {
z-index: -1;
}
	Блок з пасткою ігнорується

Як бачимо з результатів експерименту лише три варіанти приховування елементів на сторінці ігноруються програмою Apple Voice Over, а саме:

- видалення елемента з DOM сторінки – властивість display: none;

- згортання елементу за рахунок присвоєння йому нульової висоти та ширини та його приховування – властивість overflow: hidden;
- приховування елементу виставляючи його за іншими – властивість z-index: -1.

Таким чином, враховуючи результати двох експериментів, з-поміж шести запропонованих методів приховування елементів на веб-сторінці було вибрано метод згортання елементу за рахунок присвоєння нульової висоти та ширини та його приховування. Цей метод приховування елементів на веб-сторінці показав необхідні для нашої системи виявлення веб-скраперів результати, а саме був проігнорований програмою для читання екрану для людей з вадами зору Apple Voice Over та був виявлений трьома програмами для веб-скрапінгу, вибраними для проведення експерименту (FMiner, Octoparse та Visual Scraper).

Вміст файлу trap.php, що відповідає за розповсюдження пасток на всі сторінки веб-сайту виглядає наступним чином:

```
<?php
 //підключення бібліотеки для роботи з елементами HTML
 include_once('simple_html_dom.php');
 //знаходження тегу <body>
 $html = str_get_html('<body>');
 if ($html) {
  //додавання прихованого блоку з пасткою до сторінки
  $html->find('body', 0)->innertext = '
     <div style="overflow: hidden; width: 0; height: 0;">
       <a href="honeypot.html">Link for web scraper</a>
     </div>
  ';
 }
 echo $html;
```

3.4 Опис підсистеми виявлення веб-скраперів на веб-сайті

Підсистема виявлення працює наступним чином (див рис. 3.5). Коли веб-скрапер потрапляє в пастку honeypot.html, спрацьовує функція визначення IP-адреси (getUserIP()). І наступним кроком є запис визначеної IP-адреси в файл ip_list.txt. Далі з цим файлом працюють підсистеми автоматичного блокування та сповіщення.

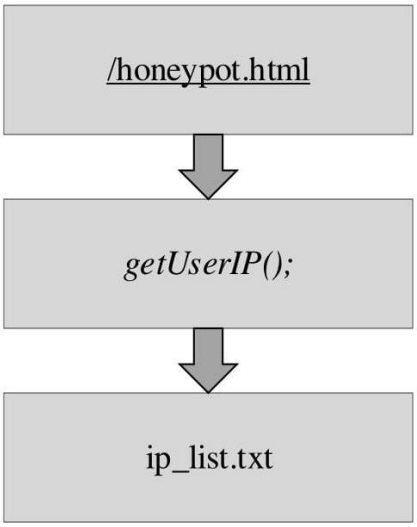

Рисунок 3.5 – Схема роботи підсистеми виявлення веб-скраперів

Нижче представлена реалізація функції getUserIP(), що визначає IP-адресу користувача чи системи, що потрапила на сторінку-пастку.

```
<?php
   function getUserIP(){
      if (isset($_SERVER['REMOTE_ADDR'])) {
         $ip = $_SERVER['REMOTE_ADDR'];
      }
      return $ip;
   }
```

Нижче представлена реалізація механізму додавання виявлених IP-адрес веб-скраперів до файлу ip_list.txt. Цей код реалізований прямо на сторінці-пастці, куди потрапляє веб-скрапер.

```
<?php
   include_once('getIP.php');
   //визначення IP-адреси
   $userIP = getUserIP();
   $ip_list = dirname(__FILE__).'/ip_list.txt';
   $blacklist = file($ip_list, FILE_IGNORE_NEW_LINES);
   //перевірка чи є така IP-адреса в файлі
   if (in_array($userIP, $blacklist))
      ;
   else {
      //запис нової IP-адреси в файл
      file_put_contents($ip_list, $userIP . PHP_EOL, FILE_APPEND);
   }
?>
```

3.5 Опис підсистеми автоматичного блокування веб-скраперів

Підсистема автоматичного блокування системи виявлення веб-скраперів з використанням пасток націлена на блокування виявлених веб-скраперів автоматично і без втручання людини. Для блокування було розглянуто два варіанти:

1. Додавання виявлених IP-адрес до файлу .htaccess, що повністю заблокує доступ з певної IP-адреси на серверному рівні;
2. Перевірка IP-адреси кожного звернення і порівняння визначеної IP-адреси з файлом ip_list.txt, за який відповідає підсистема виявлення.

Було проведено аналіз обох методів. Перший варіант є менш затратним, але в цілях безпеки категорично не рекомендується дозволяти автоматизований запис до файлу .htaccess. Другий варіант потребує більше серверних ресурсів, хоча ніякої, помітної для звичайного користувача, затримки не створює. Також реалізація автоматичного блокування виявлення IP-адрес з використанням можливостей мови програмування PHP є більш гнучким.

Нижче представлена реалізація основного функціоналу підсистеми автоматичного блокування виявлених веб-скраперів. Даний функціонал міститься у файлі deny.php, який підключається до файлу loader.php, який, в свою чергу, додається до кожної сторінку веб-сайту.

```
<?php
   include_once('getIP.php');
   $filename = dirname(__FILE__).'/ip_list.txt';
   $blacklist = file($filename, FILE_IGNORE_NEW_LINES);
   //визначаємо IP-адресу
   $userIP = getUserIP();
   //порівнюємо зі збереженими IP-адресами
   if (in_array(strval(trim($userIP)), $blacklist)) {
      if ($_SERVER['REQUEST_URI'] != '/denied.html') {
         //якщо знайдено збіг, то переадресовуємо на сторінку /denied.html
         header('Location: http://URL/denied.html');
         die();
      }
   }
```

Підсистема автоматичного блокування працює наступним чином (див. рис. 3.6).

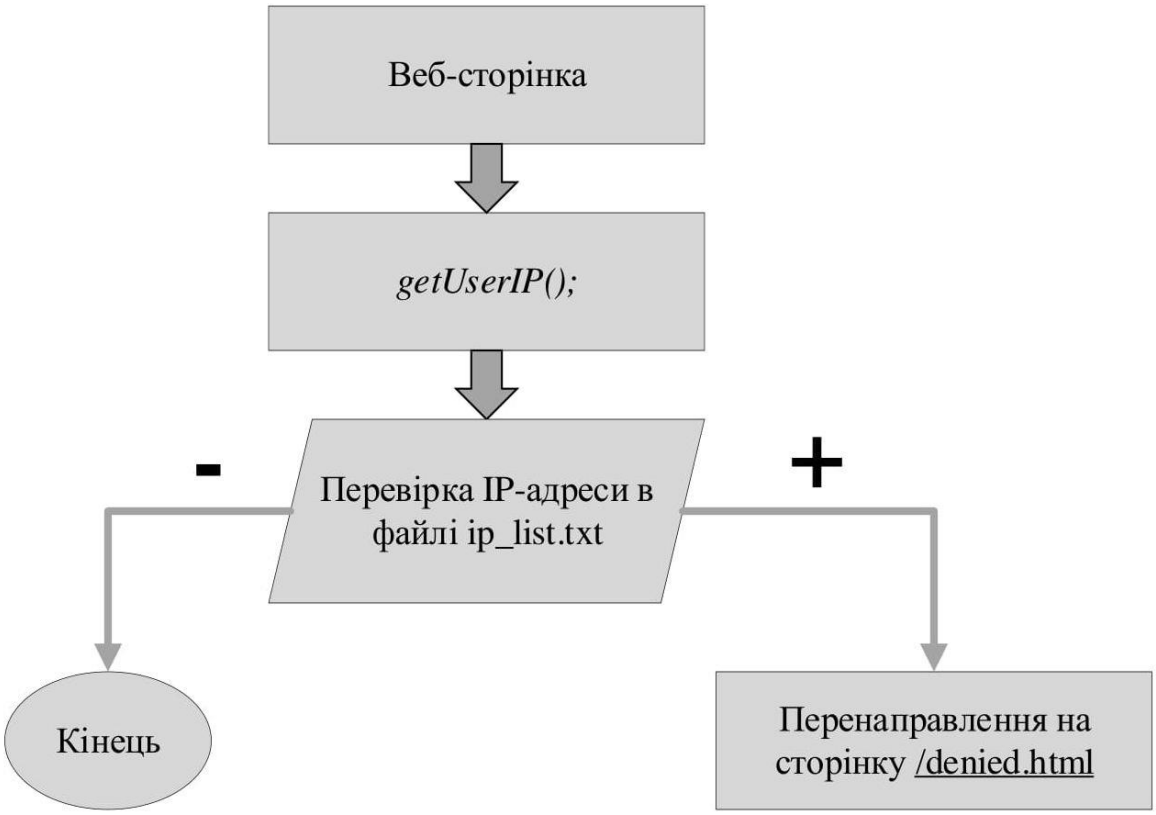

Рисунок 3.6 – Схема роботи підсистеми автоматичного блокування веб-скраперів

Спочатку користувач потрапляє на будь-яку сторінку веб-сайту. Система отримує IP-адресу, з якої здійснено звернення і перевіряє її наявність в списку виявлених веб-скраперів (ip_list.txt). Якщо такої IP-адреси не знайдено, користувач отримує доступ до сторінки. Якщо збіг знайдено, система перенаправляє запит з такої IP-адреси на сторінку /denied.html, яка була створена при підготовці системи до використання (див. рис. 3.7).

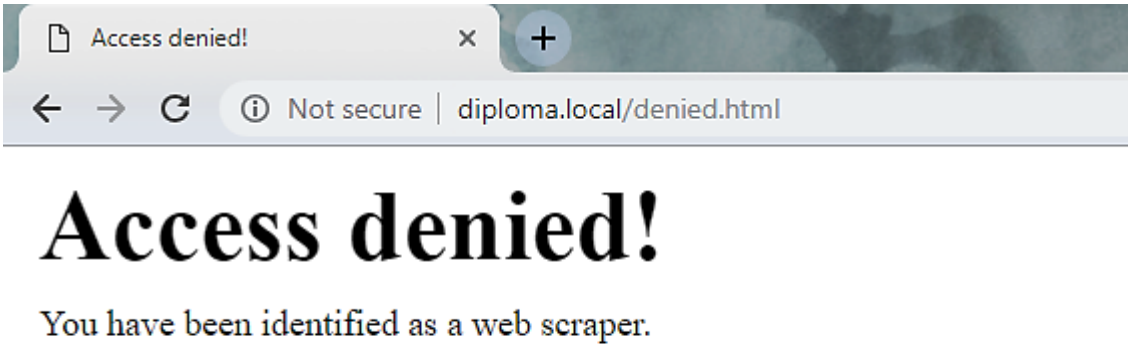

Рисунок 3.7 – Приклад сторінки /denied.html

Після блокування доступу до веб-сайту для визначеної IP-адреси починає роботу підсистема сповіщення про виявлення веб-скраперів.

3.6 Опис підсистеми сповіщення про виявлення веб-скраперів

Підсистема сповіщення працює наступним чином:

- Веб-скрапер потрапляє на сторінку-пастку
- Зчитується його адреса (робота підсистеми виявлення та підсистеми автоматичного блокування)
- Виконується функція sendBlacklistMail($ip), як отримує на вхід IP-адресу веб-скрапера і формує сповіщення, що відправляється на електронну пошту адміністратора веб-сайту.

Нижче зображена схема роботи цієї підсистеми (див. рис. 3.8).

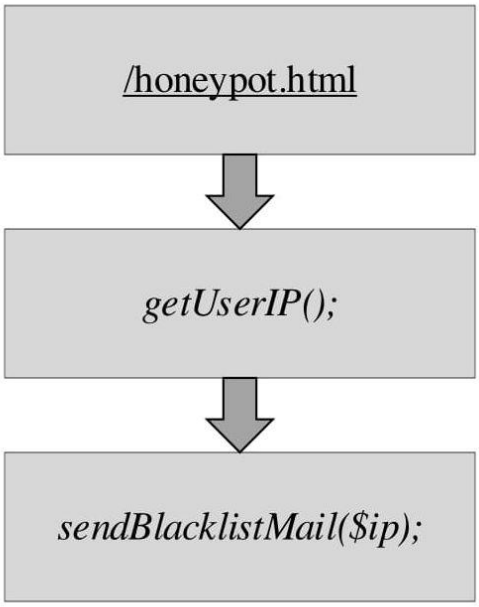

Рисунок 3.8 – Схема роботи підсистеми сповіщення

Реалізація функції автоматичного відправлення сповіщення про виявлення веб-скрапера адміністратору веб-сайту представлена нижче. Дана функція відправляє повідомлення з серверної електронної пошти на вказану в функції (див. рис. 3.9).

```
<?php
   function sendBlacklistMail($ip)
   {
       $to = '[email address]';
       $subject = 'New web scraper was detected';
       $message = "New web scraper was detected on your website. The blacklist
       was successfully updated with new IP address - " . $ip . " \r\n\r\nKeep your
       data safe and secured!\r\nPavlo Sakhanda";
       mail($to, $subject, $message);
   }
```

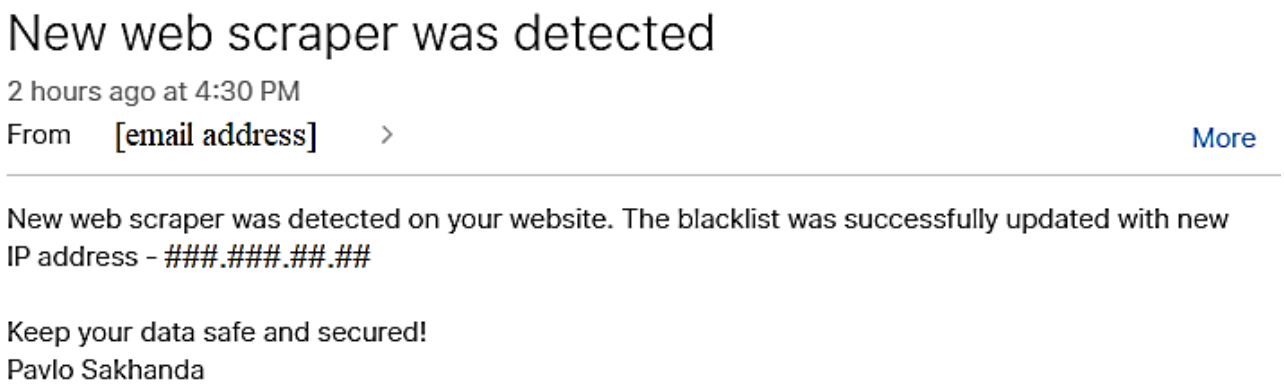

Рисунок 3.9 – Приклад повідомлення про виявлення веб-скрапера на веб-сайті

Ця підсистема є заключною ланкою циклу дій при виявленні веб-скарпера, який постійно працює в реальному часі.

3.7 Опис підсистеми моніторингу структури сайту

Останньою розглядаємо підсистему моніторингу структури сайту. Ця підсистема відповідає за внесення змін до файлу з віртуальними адресами пасток. Вона реагує на зміни в структурі сайту та інформує про це адміністратора веб-сайту. Це необхідно для створення перенаправлень з нових віртуальних пасток на одну фізичну сторінку.

Давайте розглянемо схему роботи підсистеми моніторингу структури сайту (див. рис. 3.10).

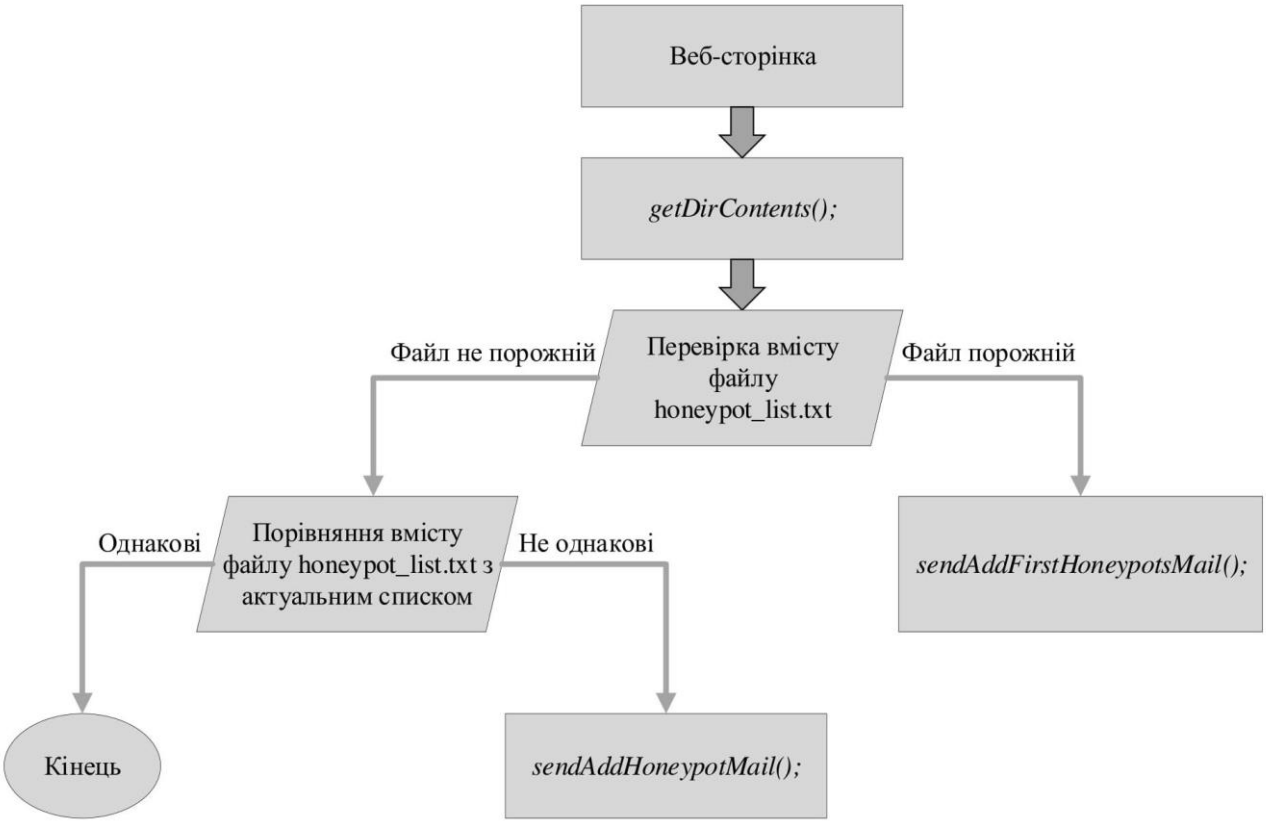

Рисунок 3.10 – Схема роботи підсистеми моніторингу структури сайту

При потраплянні на будь-яку сторінку веб-сайту спрацьовую функція getDirContents(). Реалізація цієї функції виглядає на ступним чином:

```
function getDirContents($dir) {
    $results = [];
    $files = scandir($dir);
    foreach($files as $key => $value){
        if ($value === '.' || $value === '..' || $value === '.idea') {
            continue;
        }
        $path = str_replace(DIRECTORY_SEPARATOR, '/', realpath($dir . '/' .
        $value));
        if (is_dir($path)) {
            $results[] = $path;
```

```
            foreach (getDirContents($path) as $p) {
                $results[] = $p;
            }
        }
    }
    return $results;
}
```

Основний функціонал цієї підсистеми закладений саме в цю функцію. Вона перевіряє всю структуру сайту і рекурсивно отримує список всіх директорій (див. рис. 3.11), що в подальшому необхідно для створення перенаправлень з віртуальних пасток на фізичну сторінку-пастку.

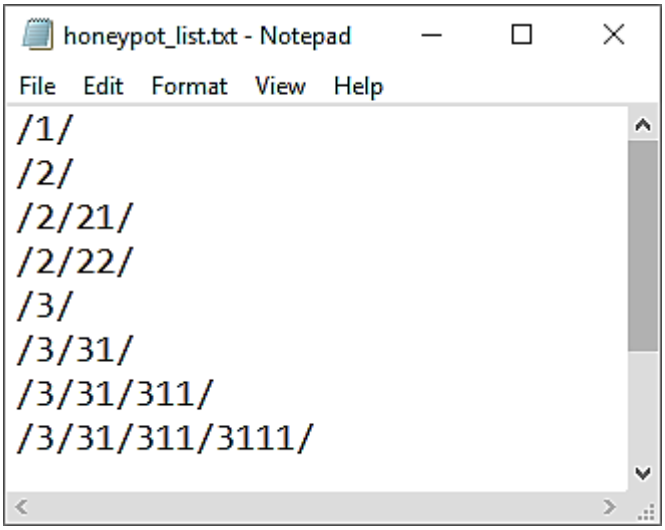

Рисунок 3.11 – Приклад файлу, що містить посилання до віртуальних пасток

Якщо файл порожній, це означає, що віртуальні пастки ще не були налаштовані і адміністратор веб-сайту отримує відповідне сповіщення (див. рис. 3.12).

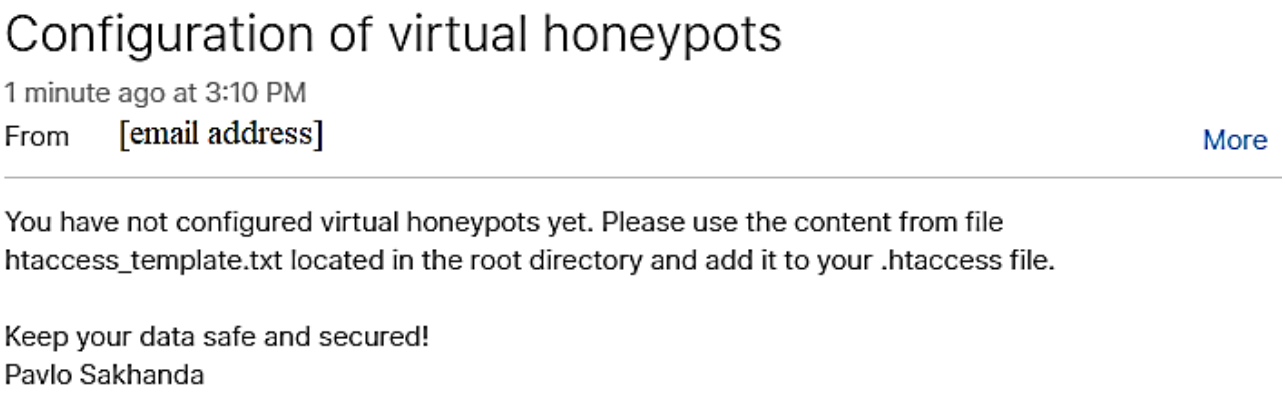

Рисунок 3.12 – Приклад повідомлення про початкове налаштування файлу .htaccess

Приклад файлу htaccess_template.txt наведено на рисунку 3.13.

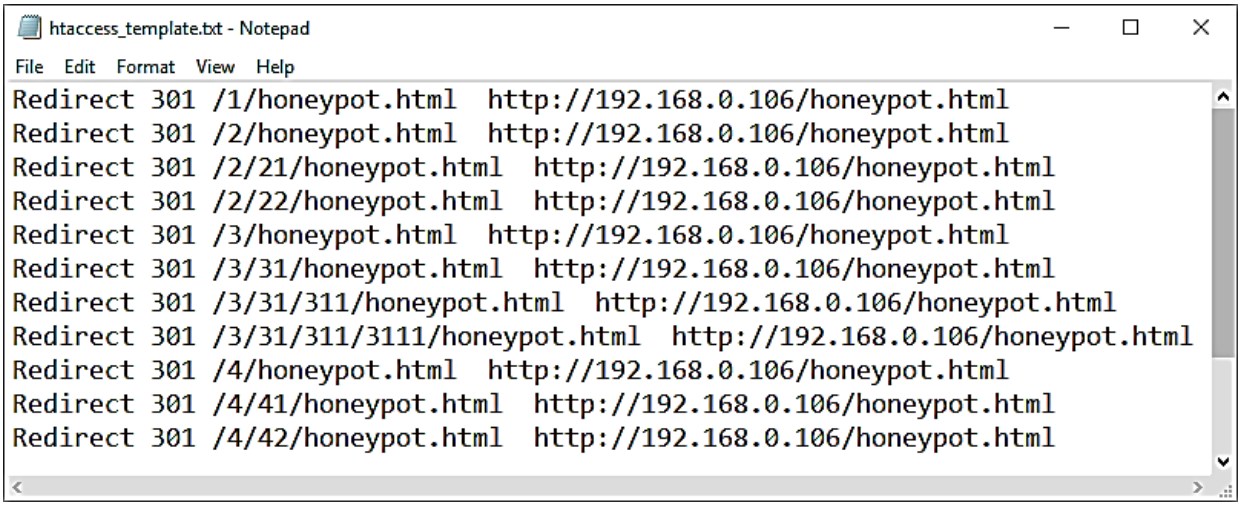

Рисунок 3.13 – Приклад файлу htaccess_template.txt

Після цього функція звіряє вміст файлу honeypot_list.txt з поточним списком директорій. Якщо є якісь відмінності, то система надсилає адміністратору сповіщення, що потрібно оновити список віртуальних пасток і їх перенаправлення (див. рис. 13).

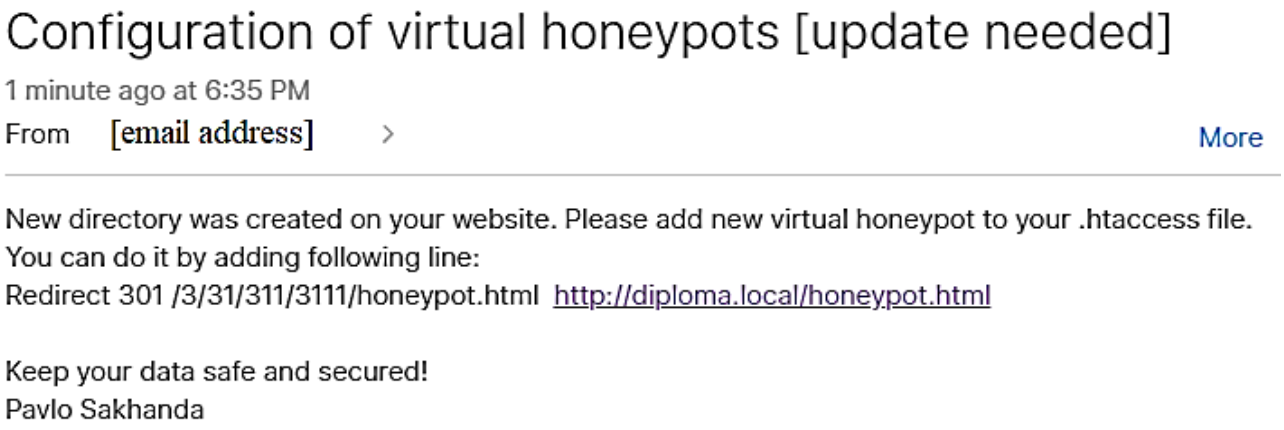

Рисунок 3.13 – Приклад повідомлення про необхідність оновлення файлу .htaccess

Це єдиний процес в системі виявлення веб-скраперів який не було повністю автоматизовано. Як вказувалось в попередніх підрозділах цього розділу, в цілях безпеки не рекомендується дозволяти автоматизований запис в файл .htaccess.

Висновки до розділу 3

В даному розділі було розглянуто архітектуру системи виявлення веб-скраперів з використанням пасток. Вона складається з п’яти підсистем:

- Підсистема розповсюдження пасток;
- Підсистема моніторингу структури веб-сайту;
- Підсистема виявлення;
- Підсистема автоматичного блокування;
- Підсистема сповіщення.

Кожна з підсистем була детально розглянута. Було продемонстровано основний функціонал кожної з підсистем та алгоритм їх роботи.

Експериментальним методом було вибрано найбільш ефективний метод приховування пасток на сторінках веб-сайту – метод згортання елементу за

рахунок присвоєння нульової висоти та ширини та його приховування (overflow: hidden).

Також було обрано найбільш безпечний метод автоматичного блокування виявлених веб-скраперів, що не потребує автоматизованого запису в файл .htaccess.

4 АНАЛІЗ РЕЗУЛЬТАТІВ

Для отримання числового значення ефективності системи та захищеності інформації було проведено декілька експериментів, які будуть описані в наступних підрозділах цього розділу.

Для доведення ефективності запропонована система виявлення з використанням удосконаленого методу представлення пасток на веб-сайті була порівняна з системами такої ж архітектури, тільки з використанням звичайного методу додавання пасток. Таких систем було спроектовано дві. В першій пастка була розміщена в кореневому каталозі, а в іншій – в будь-якому іншому.

Час виявлення веб-скрапера визначався наступним чином: при запуску програми записувався час переходу на першу сторінку, потім записувався час переходу на пастку і блокування веб-скрапера. Обчисливши різницю двох виміряних величин отримаємо час, за який веб-скрапер було виявлено системою.

Для зручності в таблицях з результатами експериментів будуть використовуватись наступні умовні позначення:

- Система №1 – Система виявлення з використанням звичайного методу додавання пасток (пастка розміщена в кореневому каталозі).
- Система №2 – Система виявлення з використанням звичайного методу додавання пасток (пастка розміщена не в кореневому каталозі).
- Система №3 – Система виявлення з використанням удосконаленого методу додавання пасток.
- Веб-сайт №1 – інтернет-магазин.
- Веб-сайт №2 – веб-сайт, що містить багато статей (інформаційний ресурс).
- Веб-сайт №3 – невеликий веб-сайт компанії.

Всі результати а таблицях 4.1 – 4.5 наведені в секундах.

4.1 Експеримент з визначенням швидкості виявлення веб-скраперів

Перший експеримент – це визначення часу, за який веб-скрапер буде виявлений системою. Для доведення ефективності запропонована система виявлення з використанням удосконаленого методу представлення пасток на веб-сайті була порівняння з системами такої ж архітектури, тільки з використанням звичайного методу додавання пасток. Таких систем було спроектовано дві. В першій пастка була розміщена в кореневому каталозі, а в іншій – в будь-якому іншому.

Час виявлення веб-скрапера визначався наступним чином: при запуску програми записувався час переходу на першу сторінку, потім записувався час переходу на пастку і блокування веб-скрапера. Обчисливши різницю двох виміряних величин отримаємо час, за який веб-скрапер було виявлено системою.

Спершу обрахуємо час, за який веб-скрапер буде виявлено при звичайному запуску програмного забезпечення з головної сторінки веб-сайту. Веб-скрапер було запущено без спеціальних налаштувань. Результати представлено в таблиці 4.1.

Таблиця 4.1 – Порівняння часу, за який буде виявлено веб-скрапер без специфічних налаштувань з використанням різних методів виявлення та в умовах старту програми з головної сторінки веб-сайту.

	Веб-сайт №1			**Веб-сайт №2**			**Веб-сайт №3**		
	1	2	3	1	2	3	1	2	3
Система №1	0,024	0,023	0,039	0,03	0,018	0,044	0,011	0,009	0,017
Система №2	0,039	0,034	0,057	0,069	0,044	0,057	0,02	0,021	0,027
Система №3	0,021	0,023	0,029	0,02	0,018	0,024	0,008	0,01	0,014

Другий дослід – запуск програми починаючи з будь-якої іншої сторінки, окрім головної. Такий дослід є необхідним через те, що досить часто практикується запуск веб-скраперів не з кореневої директорії, так як найбільш ймовірно саме там буде знаходитись пастка.

Результати наведено в таблиці 4.2.

Таблиця 4.2 – Порівняння часу, за який буде виявлено веб-скрапер без специфічних налаштувань з використанням різних методів виявлення та в умовах старту програми не з головної сторінки веб-сайту.

	Веб-сайт №1			**Веб-сайт №2**			**Веб-сайт №3**		
	1	2	3	1	2	3	1	2	3
Система №1	0,029	0,039	0,049	0,059	0,054	0,056	0,036	0,031	0,039
Система №2	0,026	0,015	0,054	0,049	0,049	0,055	0,031	0,037	0,034
Система №3	0,019	0,02	0,031	0,024	0,02	0,027	0,007	0,011	0,009

Результати будуть оброблені в наступних підрозділах.

4.2 Експеримент з налаштованими на визначену директорію веб-скраперами

В цьому експерименті веб-скрапер було налаштовано на пошук по певній директорії сайту. Тобто, якщо посилання чи ресурс не знаходиться в потрібній директорії, програма його ігнорує. Саме на такий принцип використання веб-скраперів було орієнтоване створення запропонованої системи виявлення.

Спочатку зімітуємо пошук по директорії веб-сайту, в якій нема пастки. Можна зробити припущення, що Система №1 та Система №2 пропустять таке програмне забезпечення, але це потрібно довести.

Результати показано в таблиці 4.3.

Таблиця 4.3 – Порівняння часу, за який буде виявлено веб-скрапер налаштований сканувати лише визначені директорії з умовою, що пастка в цих директоріях не знаходиться.

	Веб-сайт №1			**Веб-сайт №2**			**Веб-сайт №3**		
	1	2	3	1	2	3	1	2	3
Система №1	_	_	_	_	_	_	_	_	_
Система №2	_	_	_	_	_	_	_	_	_
Система №3	0,015	0,009	0,02	0,017	0,01	0,024	0,005	0,009	0,011

Як і припускалось, в перших двох випадках система взагалі не змогла виявити присутність веб-скрапера на веб-сайті, що зводить захищеність інформації, що знаходиться на інтернет-ресурсі до нуля.

Другий дослід – запуск програми по директорії, що містить пастку. Результати наведені в таблиці 4.4.

Таблиця 4.4 – Порівняння часу, за який буде виявлено веб-скрапер налаштований сканувати лише визначені директорії з умовою, що пастка знаходиться в одній з цих директорій.

	Веб-сайт №1			**Веб-сайт №2**			**Веб-сайт №3**		
	1	2	3	1	2	3	1	2	3
Система №1	0,037	0,033	0,034	0,041	0,026	0,041	0,013	0,025	0,026
Система №2	0,023	0,025	0,039	0,031	0,033	0,039	0,029	0,023	0,031
Система №3	0,017	0,016	0,021	0,019	0,02	0,021	0,006	0,008	0,013

Результати будуть оброблені в наступних підрозділах.

4.3 Оцінка ефективності системи

Для можливості оцінки ефективності системи та захищеності інформації потрібно опрацювати отримані результати.

Для цього експериментальним методом було отримано час, за який веб-скрапер отримує одиницю інформації. Для цього було проведено два досліди.

Перший експеримент – на веб-сторінці було розміщено блок з текстом. З використанням трьох веб-скраперів було отримано середній час, за який програма може зчитати цей блок.

Другий експеримент – не веб-сторінці було розміщено блок з зображенням. Таким самим способом було отримано середній час, за який можна автоматизовано отримати це зображення.

Результати експерименту приведені в таблиці 4.5.

Таблиця 4.5 – Середній час автоматизованого отримання одиниці інформації з веб-сайту.

	Fminer		**Octoparse**		**Visual Scraper**		**Середнє значення**
	1	2	1	2	1	2	
Текст	0,0039	0,0032	0,0042	0,0038	0,0056	0,0059	0,0044
Зображення	0,0041	0,004	0,0044	0,0041	0,0064	0,0068	0,0049

Оскільки значення дуже близькі, для зручності обчислення ефективності системи та захищеності ресурсів ми визначимо середнє значення отриманих в ході експерименту результатів і будемо називати це часом, за який веб-скрапер отримує одиницю інформації.

Тож маємо 0.00465 секунди. Саме за скільки часу веб-скрапер може отримати певну одиницю інформацію.

Ефективність системи ми будемо визначати за допомогою кількості втраченої інформації за час, поки веб-скрапер не буде виявлено. Для цього потрібно поділити час, отриманий під час експериментів з різними видами представлення пасток на середній час отримання веб-скрапером однієї одиниці інформації.

Оскільки одиниця інформації має бути цілим числом, а визначений середній час отримання одиниці інформації є значенням приблизним, то будемо заокруглювати значення відповідно до прийнятих норм:
≥0.5 => 1, <0.5 => 0.

Результати в таблицях 4.6 – 4.9 наведено в умовних одиницях інформації.

В таблиці 4.6 представлено результати першого досліду першого експерименту.

Таблиця 4.6 – Кількість втраченої інформації за час, доки веб-скрапер без специфічних налаштувань в умовах старту програми з головної сторінки веб-сайту не буде виявлено.

	Веб-сайт №1			**Веб-сайт №2**			**Веб-сайт №3**		
	1	2	3	1	2	3	1	2	3
Система №1	5	5	8	6	4	9	2	2	4
Система №2	8	7	12	15	9	12	4	5	6
Система №3	5	5	6	4	4	5	2	2	3

Отримано наступні середні значення для кожної з систем:

- Система №1 – втрачено 5 одиниць інформації.
- Система №2 – втрачено 9 одиниць інформації.
- Система №3 – втрачено 4 одиниці інформації.

В таблиці 4.7 обчислено втрати інформації для другого досліду першого експерименту.

Таблиця 4.7 – Кількість втраченої інформації за час, доки веб-скрапер без специфічних налаштувань в умовах старту програми не з головної сторінки веб-сайту не буде виявлено.

	Веб-сайт №1			**Веб-сайт №2**			**Веб-сайт №3**		
	1	2	3	1	2	3	1	2	3
Система №1	6	8	11	13	12	12	8	7	8
Система №2	6	3	12	11	11	12	7	8	7
Система №3	4	4	7	5	4	6	2	2	2

Отримано наступні середні значення для кожної з систем:

- Система №1 – втрачено 9 одиниць інформації.
- Система №2 – втрачено 8 одиниць інформації.
- Система №3 – втрачено 4 одиниці інформації.

В таблиці 4.8 обчислено втрати інформації для першого досліду другого експерименту.

Таблиця 4.8 – Кількість втраченої інформації за час, доки веб-скрапер налаштований сканувати лише визначені директорії з умовою, що пастка в цих директоріях не знаходиться не буде виявлено.

	Веб-сайт №1			**Веб-сайт №2**			**Веб-сайт №3**		
	1	2	3	1	2	3	1	2	3
Система №1	100%	100%	100%	100%	100%	100%	100%	100%	100%
Система №2	100%	100%	100%	100%	100%	100%	100%	100%	100%
Система №3	3	2	4	4	2	5	1	2	2

Маємо наступні результати:

- Система №1 – веб-скрапери не були виявлені, вся потрібна інформація була отримана
- Система №2 – веб-скрапери не були виявлені, вся потрібна інформація була отримана
- Система №3 – втрачено 3 одиниці інформації.

В таблиці 4.9 обчислено втрати інформації для другого досліду другого експерименту.

Таблиця 4.9 – Кількість втраченої інформації за час, доки веб-скрапер налаштований сканувати лише визначені директорії з умовою, що пастка знаходиться в одній з цих директорій не буде виявлено.

	Веб-сайт №1			**Веб-сайт №2**			**Веб-сайт №3**		
	1	2	3	1	2	3	1	2	3
Система №1	8	7	7	9	6	9	3	5	6
Система №2	5	5	8	7	7	8	6	5	7
Система №3	4	3	5	4	4	5	1	2	3

Отримано наступні середні значення для кожної з систем:

- Система №1 – втрачено 7 одиниць інформації.
- Система №2 – втрачено 7 одиниць інформації.
- Система №3 – втрачено 3 одиниці інформації.

Як бачимо з результатів дослідження, в деяких випадках системи виявлення з використанням звичайних пасток та з використанням удосконаленого методу ведуть себе практично однаково (перший дослід першого експерименту).

В деяких випадках, результати системи з використанням удосконаленого методу розповсюдження пасток трохи краще (другі досліди кожного з експериментів).

Це зумовлено тим, що веб-скрапери архітектурно запрограмовані обходити сайт ієрархічно. Саме через те, що пастка завжди знаходиться в тій самій директорії, що і веб-скрапер, в експериментах, коли веб-скрапер мав потрапити на пастку за приблизно однаковий час, він знаходив пастку швидше під час використання удосконаленого методу розповсюдження пасток.

Та в випадку, коли пастка відсутня в директорії, на яку націлений веб-скрапер, Система №1 та Система №2 взагалі не можуть виявити веб скрапер, що повністю доводить ефективність системи з використанням удосконаленого методу розповсюдження пасток. Також це доводить досягнення мети даної роботи – підвищення захищеності інформації, що знаходиться на веб-сайті.

Результати проілюстровані на рисунку 4.1.

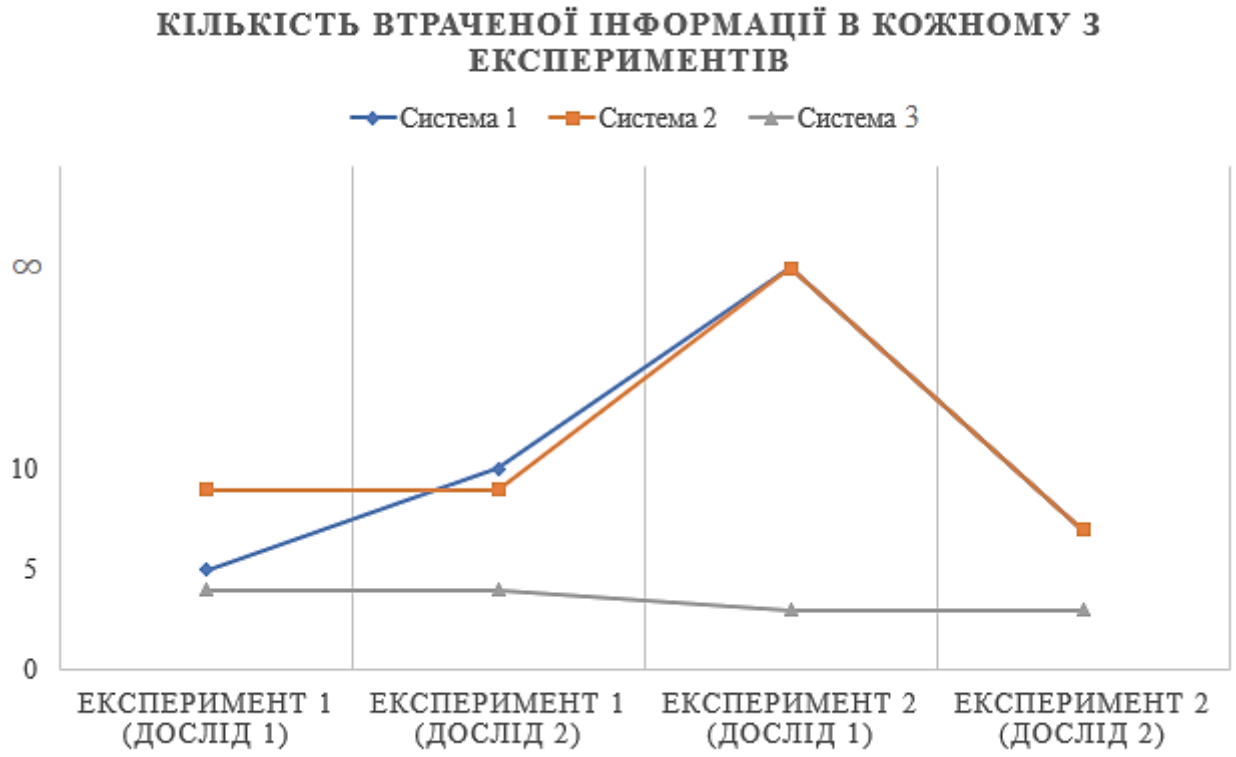

Рисунок 4.1 – Кінцеві результати експериментів, що показують середню кількість втраченої інформації

Також слід зазначити, що не звертаючи увагу на процеси, які відбуваються під час роботи системи, час виявлення веб-скрапера для

випадків, коли системи з використанням звичайних пасток та з використанням удосконаленого методу були в однакових умовах не збільшився, що ще раз доводить ефективність системи за рахунок стовідсоткового покриття веб-сайту пастками без завантаження серверу та не знижуючи продуктивність.

4.4 Рекомендації щодо впровадження системи виявлення веб-скраперів з використанням пасток

Система виявлення була розроблена для веб-сервера Apache, але в другому розділі були представлені варіанти виконання тих самих функцій і на інших веб-серверах (IIS, nginx). Дану систему можна пристосувати до будь-якого веб-серверу, що може виконувати PHP-код.

Для правильної роботи системи необхідна версія PHP не нижче 5.6. Рекомендується використовувати PHP версії 7.

Поміж стандартних процедур, що необхідні для використання системи (див. підрозділ 3.2), необхідно дотримуватись деяких правил під час використання запропонованої системи виявлення веб-скраперів:

- В роботі використовуються назви файлів та класів для наочності та кращого розуміння. Не рекомендується використовувати такі слова як honeypot, trap. Система захисту веб-скрапера може передбачити такі методи виявлення і додати такі назви в список виключень.
- В цілях безпеки не рекомендується автоматизовувати запис в .htaccess файл для повної автоматизації роботи системи виявлення. Один ручний процес, який вимагає лише копіювання та вставки в момент, коли про це сповістить система не є критичним.
- Рекомендується приховати доступ до списку всіх папок та файлів на сервері для заборони копіювання створених PHP-скриптів. Це робиться за допомогою додавання наступного рядка до файлу конфігурацій .htaccess: Options -Indexes.

Висновки до розділу 4

В даному розділі були розглянуті основні результати досліджень. Було доведено ефективність системи та підвищення захищеності інформації, що знаходиться на веб-сайтах. Для цього в рамках дослідження було проведено два експерименти з використанням різних веб-скраперів.

Порівнювалась робота системи виявлення з використанням удосконаленого методу приховування пасток та звичайні методи їх додавання. В жодному з експериментів створена система не показала гірший результат за існуючі архітектурні рішення, що доводить, що даних метод покриває пастками весь сайт без використання зайвих ресурсів.

В деяких варіантах було продемонстровано трохи кращі результати через архітектуру веб-скраперів, що обходять веб-сайт ієрархічно по директоріям.

Та найголовнішим результатом було виявлення веб-скраперів запропоноваю системою в умовах, коли система з використанням звичайного методу приховування пасток не змогла виявити програму. Таким чином створена система продемонструвала не тільки кращі результати виявлення веб-скраперів, а й кращі показники захищеності інформації.

ВИСНОВКИ

В даній магістерській дисертації були детально розглянуті та проаналізовані такі поняття як веб-скрапер та веб-скрапінг. Було визначено сфери їх використання та загрози, які несе веб-скрапінг.

В ході аналізу було виділено наступні ситуації, коли веб-скрапінг може стати незаконним:

- Веб-сайт, включаючи його сторінки, дизайн, макет і базу даних – можуть бути захищені авторським правом, оскільки він розглядається як творча робота. Вилучення з нього даних і простий факт копіювання веб-сторінки за допомогою веб-скрапера може вважатися порушенням авторських прав.
- Використання статей та будь якого іншого авторського матеріалу у власних цілях
- Отримання персональних даних людей, для створення клієнтських баз даних та подальшого їх використання
- Порушення Правил використання, навіть якщо файл robots.txt не забороняє веб-скрапінг на веб-сайті.

Також в роботі були розглянуті найбільш поширені методи виявлення та блокування веб-скраперів. Були проаналізовані різні види пасток та методи їх представлення на веб-сторінці.

Для досягнення найкращого результату, експериментальним методом було виявлено найбільш ефективний метод приховування пасток на сторінках веб-сайту – метод згортання елементу за рахунок присвоєння нульової висоти та ширини та його приховування (overflow: hidden).

Аналізуючи існуючі методи, що широко використовуються було розроблене вдосконалення методу виявлення веб-скраперів з використанням пасток. Запропонований метод забезпечує стовідсоткове покриття всіх сторінок веб-сайту пастками і при цьому не потребує створення чи

використання додаткових сторінок. Ефективність методу полягає в виявленні тих веб-скраперів, що сканують веб-сайт лише по певних директоріях. Таким чином, якщо веб-скрапер опрацьовує лише ті директорії сайту, які йому потрібні, він в будь-якому разі потрапить в одну з пасток.

Було розроблено систему виявлення веб-скраперів з використанням пасток. Вона складається з п’яти підсистем:

- Підсистема розповсюдження пасток;
- Підсистема моніторингу структури веб-сайту;
- Підсистема виявлення;
- Підсистема автоматичного блокування;
- Підсистема сповіщення.

Основою системи став розроблений вдосконалений метод виявлення веб-скраперів з використанням пасток.

Для виконання поставлених завдань та досягнення мети даної роботи було проведено ряд експериментів.

Порівнювалась робота системи виявлення з використанням удосконаленого методу приховування пасток та звичайні методи їх додавання. В жодному з експериментів створена система не показала гірший результат за існуючі архітектурні рішення, що доводить, що даних метод покриває пастками весь сайт без використання зайвих ресурсів.

В деяких варіантах було продемонстровано трохи кращі результати через архітектуру веб-скраперів, що обходять веб-сайт ієрархічно по директоріям.

Також слід зазначити, що не звертаючи увагу на процеси, які відбуваються під час роботи системи, час виявлення веб-скрапера для випадків, коли системи з використанням звичайних пасток та з використанням удосконаленого методу були в однакових умовах не збільшився, що ще раз доводить ефективність системи за рахунок стовідсоткового покриття веб-сайту пастками без завантаження серверу та не знижуючи продуктивність.

Все ж найголовнішим результатом було виявлення веб-скраперів запропоноваю системою в умовах, коли система з використанням звичайного методу приховування пасток не змогла виявити програму. Таким чином створена система продемонструвала не тільки кращі результати виявлення веб-скраперів, а й кращі показники захищеності інформації.

Можна зазначити, що результати досліджень доводять ефективність системи, підвищення рівня захищеності інформації, що знаходиться на веб-сайтах та досягнення поставленої мети.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Web Scraping Tools to Extract Online Data [Електронний ресурс]. – 2019 – Режим доступу до ресурсу: https://www.hongkiat.com/blog/web-scraping-tools/
2. A Comparative Study on Web Scraping [Електронний ресурс]. – 2015 – Режим доступу до ресурсу: http://ir.kdu.ac.lk/bitstream/handle/345/1051/com-059.pdf?sequence=1&isAllowed=y
3. Olston, C. and Najork, M. Web crawling. 2010. – Foundations and Trends in Information Retrieval 3:175-246.
4. Web Content Mining Techniques: A Survey [Електронний ресурс]. – 2012 – Режим доступу до ресурсу: https://research.ijcaonline.org/volume47/number11/pxc3880266.pdf
5. What are the Different Scraping Techniques [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://www.shieldsquare.com/what-are-the-different-scraping-techniques/
6. Types of Web Scraping Tools [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://medium.com/prowebscraper/types-of-web-scraping-tools-940f824622fb
7. The dangers of web scraping [Електронний ресурс]. – 2016 – Режим доступу до ресурсу: https://www.information-age.com/dangers-web-scraping-123461971/
8. Legality of Web Scraping and Crawling [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://urlzs.com/3n4QC
9. Krotov, Vlad & Silva, Leiser. (2018). Legality and Ethics of Web Scraping.
10. Protection Against Web Scraping [Електронний ресурс]. – 2017 – Режим доступу до ресурсу: https://blog.jscrambler.com/protect-your-site-against-web-scraping/

11. Doran, D and Gokhale, S. S. Web robot detection techniques: overview and limitations. 2011. – Data Mining and Knowledge Discovery 22:183-210.
12. Doran, D. Detection, Classification, and Workload Analysis of Web Robots. – 2014. – Available at: http://digitalcommons.uconn.edu/ dissertations/348.
13. Detecting and Blocking Site Scraping Attacks [Електронний ресурс]. – 2014 – Режим доступу до ресурсу: https://www.imperva.com/docs/gated/WP_Detecting_and_Blocking_Site_Scraping_Attacks.pdf
14. Dryer, A. J. and Stockton, J. Internet 'Data Scraping': A Primer for Counseling Clients. 2013. – New York Law Journal 15:1-3.
15. Mitchell, R. Web Scraping with Python: Collecting Data from the Modern Web. S. : O'Reilly Media, 2015.
16. Саханда П. П., Ткач В. М. Удосконалений метод виявлення веб-скраперів з використанням пасток. 2019 – Матеріали XVII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики»: 177-179
17. 2019 Bot Report Executive Summary [Електронний ресурс]. – 2019 – Режим доступу до ресурсу: https://resources.distilnetworks.com/white-paper-reports/2019-bot-report-executive-summary