Methodologies for Retrieving and Processing Information from Open Sources (OSINT)

Călin Ioan JULAN and Mihai TOGAN

Abstract—**Open-source intelligence (OSINT) is the collection and analysis of data gathered from open sources (covert and publicly available sources) to produce actionable intelligence. OSINT is primarily used in national security, law enforcement and business intelligence functions and is of value to analysts who use non-sensitive intelligence in answering classified, unclassified, or proprietary intelligence requirements across the previous intelligence disciplines [1]. A main hindrance to practical OSINT is the volume of information it has to deal with ("information explosion"). The amount of data being distributed increases at such a rate that it becomes difficult to evaluate sources in intelligence analysis. To a small degree, the work has sometimes been done by amateur crowdsourcing [2]. In this article, we will present the results obtained from running various open-source intelligence, gathering custom-made applications developed using C# as primary language and React/JavaScript. At the business logic layer, an artificial intelligence algorithm is presented that is able to classify and extract relevant information from all open-source links presented by the Data Access layer. Further, relational and non-relational databases are used to store data and ensure its persistence for future queries. Accredited journalists have some protection when asking questions and researching for recognized media outlets. Even so, they can be imprisoned, even executed, for seeking out OSINT. Private individuals illegally collecting data for a foreign military or intelligence agency is considered espionage in most countries. Of course, espionage that is not treason (i.e. betraying one's country of citizenship) has been a tool of statecraft since ancient times [3]. As a result, the modern technologies used in the development of the applications, the innovative character of the artificial intelligence algorithms and the feasibility/scalability of the proposed solution will, for sure, grant a new view of the OSINT domain and contribute to its importance.**

Index Terms—**HTML-Hypertext Markup Language, OSINT-Open Source Intelligence.**

I. Introduction

Open Source Intelligence (OSINT) refers to the practice of gathering information from publicly available sources for intelligence and analytical purposes. It involves the systematic collection, analysis, and interpretation of data obtained from a wide range of openly accessible sources, including websites, social media platforms, public records, news articles, and more. The scope of OSINT extends across various domains, including intelligence analysis, law enforcement, cybersecurity, corporate investigations, and journalism.

OSINT provides valuable insights into a diverse range of subjects, such as geopolitical events, security threats, competitive intelligence, market research, and public sentiment analysis. It allows analysts and researchers to access information that is openly accessible but may not be readily apparent or easily discoverable. By leveraging OSINT, professionals can gather and analyze information from multiple sources to gain a comprehensive understanding of a specific topic, individual, organization, or event.

The sources used in OSINT can be both digital and physical. Digital sources include websites, blogs, social media platforms, online forums, and public databases, while physical sources encompass newspapers, magazines, books, public speeches, conferences, and other tangible mediums of information dissemination [2].

The primary objective of OSINT is to collect, evaluate, and analyze open-source data to generate actionable intelligence and support decision-making processes. OSINT methodologies involve systematic data collection, verification, analysis, and dissemination of information in a structured and organized manner. It employs various techniques such as web scraping, data mining, natural language processing, geospatial analysis, and data visualization to extract meaningful insights from the collected data.

The scope of OSINT is continually expanding due to advances in technology, the proliferation of digital information, and the increasing interconnectedness of the world. It has become an indispensable tool in the arsenal of intelligence agencies, law enforcement agencies, cybersecurity professionals, journalists, and other entities that rely on accurate and timely information for their operations.

Overall, the definition and scope of OSINT encompass the collection and analysis of publicly available information from diverse sources, enabling organizations and individuals to gain valuable intelligence, make informed decisions, and stay ahead in an ever-changing landscape [4].

A. Data Collection

Data collection is a crucial step in the process of retrieving and processing information from open sources (OSINT). It involves identifying and selecting relevant open sources that contain the desired information. The effectiveness of OSINT depends on the ability to identify the right sources that provide accurate, reliable, and up-to-date data.

Identifying relevant open sources requires a systematic approach that considers the specific objectives of the OSINT project or investigation. Here are a few examples of how

C. I. JULAN is with the Military Technical Academy “Ferdinand I”, Bucharest, Romania (e-mail: julan_calin@yahoo.com).
M. TOGAN is with the Military Technical Academy “Ferdinand I”, Bucharest, Romania.

relevant open sources can be identified:

1. Websites and Online Platforms:

   • News websites: Websites of reputable news organizations provide valuable information on current events, politics, business, and more.

   • Government websites: Government portals and agency websites offer access to official reports, public records, legislation, and statistical data.

   • Social media platforms: Social media platforms such as Twitter, Facebook, and LinkedIn can provide insights into public sentiment, discussions, and user-generated content.

2. Public Databases and Archives:

   • Academic databases: Platforms like JSTOR, Google Scholar, and PubMed host scholarly articles, research papers, and publications across various disciplines.

   • Public records: Government databases, court records, property records, and patent databases can offer valuable information about individuals, organizations, and legal proceedings.

   • Open data portals: Many governments and organizations provide access to datasets on demographics, crime statistics, public health, and more.

3. Online Forums and Communities:

   • Specialized forums: Niche forums and discussion boards cater to specific topics, industries, or hobbies, where experts and enthusiasts share knowledge and experiences.

   • Q&A platforms: Websites like Quora and Stack Exchange host questions and answers on a wide range of subjects, providing insights into specific issues or topics.

4. Blogs and Personal Websites:

   • Subject matter experts: Blogs maintained by experts, researchers, and professionals in a specific field often offer valuable insights, analysis, and perspectives.

   • Personal websites: Individuals and professionals sometimes share relevant information or research findings on their personal websites [5].

5. Industry-specific Sources:

   • Trade publications: Magazines, journals, and online publications focused on specific industries often provide industry news, trends, and analysis.

   • Regulatory bodies: Organizations such as the Securities and Exchange Commission (SEC) or Food and Drug Administration (FDA) provide public filings, regulations, and industry-related information.

II. Automating Data Collection through WEB Scrapping and APIs

Automating data collection through web scraping and APIs (Application Programming Interfaces) is a powerful approach used in Open Source Intelligence (OSINT) to efficiently retrieve information from various online sources. This method allows for systematic and structured data extraction, enabling analysts to gather large volumes of data quickly and consistently. Here's a brief explanation of web scraping and APIs in the context of data collection for OSINT [4]:

1. Web Scraping: Web scraping involves extracting data from websites by parsing and analyzing their HTML structure. It enables automated data collection from multiple web pages, saving time and effort compared to manual data extraction. Here are the key steps involved in web scraping:

   • Identifying target websites: Determine the websites that contain the desired information relevant to the OSINT project.

   • Inspecting website structure: Analyze the HTML structure of the target web pages to identify the specific data elements (e.g., text, images, tables) to be scraped.

   • Writing scraping code: Use programming languages like Python and libraries such as Beautiful Soup, Scrapy, or Selenium to write scripts that automate the extraction of data from web pages.

   • Data extraction: Execute the scraping code to retrieve the desired data, following the defined scraping rules and patterns.

   • Data cleaning and organization: Process the scraped data to remove noise, duplicate entries, and irrelevant information. Structure the data in a consistent format for further analysis.

   • Web scraping can be an effective method for collecting information from websites that do not provide APIs or structured data formats. However, it is essential to adhere to legal and ethical guidelines while web scraping, respect the website's terms of service, and avoid excessive or disruptive requests that may impact the site's performance.

2. APIs: APIs provide a standardized and controlled way to access and retrieve data from online platforms and services. Many websites and online services offer APIs to allow developers to programmatically interact with their data. Here's how APIs are utilized for data collection in OSINT:

   • Identifying APIs: Determine if the target website or platform provides an API and whether it offers access to the desired data.

   • API authentication: Obtain the necessary API keys or access tokens to authenticate and authorize access to the API.

   • API requests and responses: Use programming languages and libraries to send requests to the API endpoints and receive responses containing the requested data.

   • Data extraction and processing: Extract relevant information from the API responses and process it according to the project requirements.

APIs provide a more structured and controlled approach to data collection compared to web scraping since they are designed to deliver data in a consistent format. They often offer access to real-time data, enabling analysts to gather the most up-to-date information. However, APIs are subject to usage limits and may require authentication or subscription fees, depending on the platform [6].

Combining web scraping and API-based data collection techniques can provide a comprehensive approach to OSINT data gathering, allowing analysts to leverage both structured data sources and unstructured web pages. The choice between web scraping and APIs depends on the availability of APIs, the nature of the target sources, and the specific requirements of the OSINT project.

III. Examination of the Most Important Elements of WEB Scraping through APIs

Some of the most important and crucial components necessary in order to be able to collect data from HTML structure are explained in the following chapter, e.g. “in Table I”.

TABLE I. WEB SCRAPING COMPONENTS

Title	Type
Api Requests	Web Request
Api authorization	Authorization
Data extraction	Data
HTML Structure	HTML

API requests are a fundamental part of interacting with web APIs. They allow you to send HTTP requests to a server hosting the API and receive a response containing the requested data or perform certain actions. Here's an overview of API requests [7]:

1. HTTP Methods/Verbs: API requests are typically made using different HTTP methods or verbs that indicate the type of action you want to perform. The most common methods used in API requests are:
   - GET: Retrieves data from the server. It is used to retrieve information and does not modify any data on the server.
   - POST: Sends data to the server to create a new resource. It is often used to submit data, such as form submissions or creating new records.
   - PUT: Updates an existing resource on the server. It replaces the entire resource with the new data.
   - PATCH: Updates a portion of an existing resource. It is used when you only want to modify specific fields of a resource.
   - DELETE: Removes a resource from the server.
2. URL/Endpoint: The URL or endpoint specifies the location where the API resides and the specific resource or data you want to interact with. It typically follows a specific pattern and may include parameters or identifiers to specify the exact data you are interested in.
3. Request Headers: API requests may include headers, which provide additional information about the request or control its behavior. Common headers include "Content-Type" to specify the format of the request payload (e.g., JSON, XML) and "Authorization" to authenticate the request.
4. Request Parameters/Payload: Depending on the API, you may need to pass parameters or data in the request payload. These parameters can be sent as query parameters in the URL for GET requests or included in the body for POST, PUT, PATCH, and DELETE requests. Parameters can be used to filter data, specify pagination, or provide additional information to the API.
5. Response: After sending an API request, the server processes the request and returns a response. The response typically includes an HTTP status code that indicates the success or failure of the request (e.g., 200 for success, 404 for not found, 500 for server error). The response may also include the requested data, error messages, or other relevant information.
6. Error Handling: API requests can encounter errors due to various reasons such as invalid input, authentication issues, or server problems. It's important to handle and process these errors appropriately by checking the status codes and response content. Error responses often include specific error codes or messages that can help you understand the issue.
7. Libraries and Tools: Many programming languages provide libraries and frameworks that simplify making API requests. For example, in Python, you can use libraries like Requests, aiohttp, or the built-in **http.client** module to send HTTP requests and handle responses.

It's crucial to refer to the API documentation to understand the specific request format, required headers, parameters, and any authentication requirements. Additionally, consider handling rate limits, implementing retries, and ensuring proper error handling to build robust and reliable API request code [7].

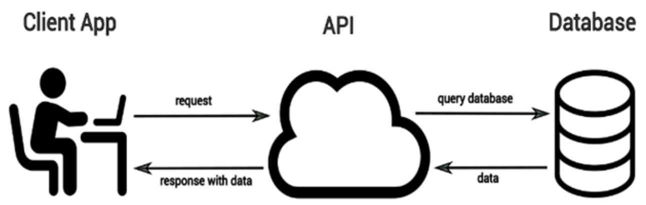

Figure 1. API Request

API authorization is the process of granting or validating the access rights of a user or an application to interact with an API. It ensures that only authorized users or applications can access and perform actions on the protected resources provided by the API. Authorization mechanisms are typically implemented as part of the API security framework and play a vital role in protecting sensitive data and maintaining the integrity of the system. Here are some common API authorization methods:

1. API Keys: API keys are unique tokens issued to developers or applications to authenticate their requests. They are typically sent as part of the request headers or query parameters. API keys act as a simple form of access control, allowing the API provider to track and monitor the usage of their API. However, API keys alone may not provide robust security and are more suitable for public APIs or low-security scenarios.
2. OAuth: OAuth (Open Authorization) is an industry-standard authorization protocol widely used for granting access to web APIs. It allows users or applications to obtain limited access rights to protected resources without sharing their actual credentials (e.g., username and password) with the API. OAuth involves an authentication step followed by the issuance of an access token, which is used to authorize subsequent API requests. OAuth provides a more secure and granular access control mechanism, allowing users to grant specific permissions (scopes) to applications.
3. JSON Web Tokens (JWT): JWT is a self-contained token format commonly used for authentication and authorization in web APIs. It consists of three parts: a header, a payload, and a signature. The payload contains claims (key-value pairs) that represent the user’s or application’s identity and other additional information. JWTs are digitally signed by the server, ensuring their integrity. They are often used as access tokens and can carry authorization information such as user roles or permissions.

4. Bearer Tokens: Bearer tokens are tokens issued by the server upon successful authentication, and they are used to authorize API requests. Bearer tokens are typically included in the "Authorization" header of the request as "Bearer <token>". They provide a straightforward way to authenticate and authorize API requests. However, securing bearer tokens is essential, as anyone possessing a valid token can access the protected resources.
5. Role-Based Access Control (RBAC): RBAC is an access control mechanism that assigns permissions and privileges based on predefined roles. In the context of API authorization, RBAC is often used to define roles such as "admin," "user," or "guest." Users or applications are assigned specific roles, and API access is determined based on those roles. RBAC provides a scalable and flexible approach to manage access rights in complex API systems.
6. API Permissions and Scopes: APIs often implement fine-grained permissions or scopes to control access to specific resources or actions. Permissions define what operations a user or application is allowed to perform, while scopes define the specific data or functionality that can be accessed. By defining and managing permissions and scopes, API providers can ensure that users or applications have the appropriate level of access to resources.

API authorization mechanisms are typically documented by the API provider, and developers need to understand and implement the required authorization method based on the API's specifications. It's important to follow security best practices, keep access tokens and credentials secure, and regularly review and audit the authorization mechanisms to maintain a secure API ecosystem.

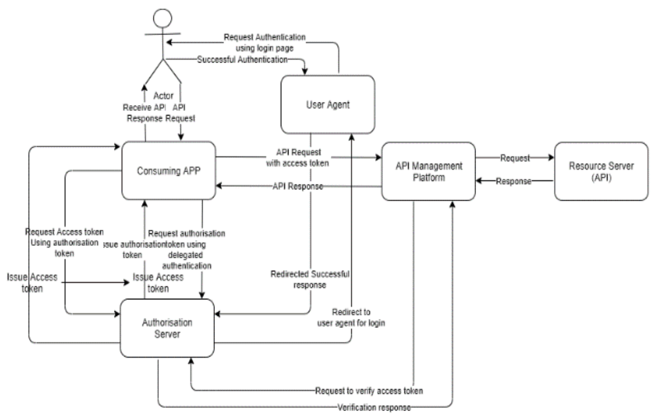
Request Authentication using login page
Successful Authentication
Actor
Receive API Response
API Request
User Agent
Consuming APP
API Request with access token
API Response
API Management Platform
Request
Response
Resource Server (API)
Request Access token Using authorisation token
Issue authorisation token
Request authorisation token using delegated authentication
Issue Access token
Issue Access token
Redirected Successful response
Redirect to user agent for login
Authorisation Server
Request to verify access token
Verification response

Figure 2. API Authorization

Data extraction and processing are crucial steps in web scraping for OSINT (Open Source Intelligence) applications. OSINT refers to the collection, analysis, and utilization of publicly available information from various online sources. Here's an overview of data extraction and processing for web scraping in OSINT applications[8]:

1. Identifying Target Websites: Determine the websites or online sources that contain the desired information for your OSINT application. These can include news sites, social media platforms, forums, blogs, public databases, government websites, and more.
2. HTML Parsing: Web scraping typically involves parsing the HTML structure of web pages to extract relevant data. You can use libraries or frameworks like BeautifulSoup (Python), JSoup (Java), or Cheerio (JavaScript) to navigate and extract data from the HTML.
3. Locating Data Elements: Analyze the structure of the target website's HTML to identify the specific elements (such as tags, classes, IDs) that contain the data you need. These elements might include article titles, author names, timestamps, content, URLs, user profiles, comments, or any other relevant information.
4. Extracting Data: Use the selected HTML elements and their corresponding selectors to extract the desired data from the web pages. You can retrieve text, links, images, tables, or any other data present in the identified elements. Apply appropriate extraction techniques like regular expressions, XPath, or CSS selectors to locate and capture the data accurately.
5. Handling Pagination and Dynamic Content: Some websites may display data across multiple pages or load content dynamically through JavaScript. You need to handle pagination by programmatically navigating through pages or modifying URLs to access all the desired data. For dynamically loaded content, you might need to use headless browsers like Puppeteer (JavaScript) or frameworks like Selenium (Python) to interact with the web page and extract the data.
6. Data Cleaning and Transformation: Raw data obtained from web scraping often requires cleaning and transformation to ensure its quality and usability. This involves removing unwanted characters, normalizing data formats, handling missing values, and removing duplicates. Data cleaning techniques such as filtering, validation, and standardization may be necessary depending on the specific OSINT application.
7. Natural Language Processing (NLP): In many OSINT applications, textual data plays a significant role. NLP techniques can be applied to process and analyze text data extracted from web pages. This can include tasks such as sentiment analysis, named entity recognition, topic modeling, keyword extraction, or language detection. NLP libraries like NLTK (Python), spaCy (Python), or Stanford NLP can assist in performing these tasks.
8. Storage and Analysis: The extracted and processed data needs to be stored for further analysis and utilization. You can save the data in various formats, such as CSV, JSON, or a database (e.g., SQLite, MySQL). Consider the specific requirements of your OSINT application and choose an appropriate storage method. Once stored, you can apply data analysis, visualization, or machine learning techniques to derive insights and make informed decisions based on the collected information.

Remember to respect the terms of service and any legal restrictions of the websites you scrape, and ensure compliance with relevant laws and regulations governing data collection and usage [9].

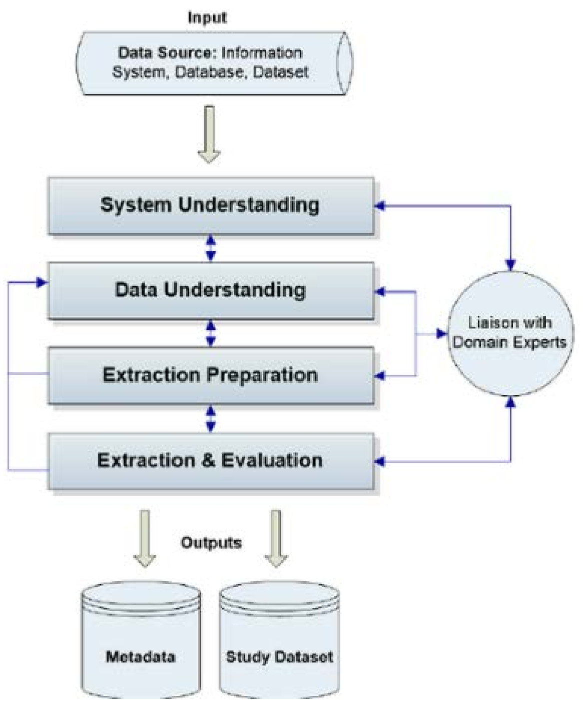

Figure 3. Data processing and extraction

HTML (Hypertext Markup Language) is the standard markup language used for creating web pages. It defines the structure and presentation of content on a web page. The HTML structure consists of various elements and tags that define the different parts and components of a web page. Here are the key components of HTML structure:

1. Tags: HTML uses tags to enclose and define elements within a web page. Tags are represented by angle brackets ("<" and ">") and can be either opening tags or closing tags. Opening tags indicate the start of an element, while closing tags indicate the end. For example, the opening tag "<p>" represents a paragraph element, and the closing tag "</p>" denotes the end of the paragraph.
2. Elements: Elements are the fundamental building blocks of HTML and represent different parts of a web page's content. Elements are defined by tags and can include text, images, links, headings, lists, forms, tables, and more. Each element has its specific purpose and behavior.
3. Attributes: HTML elements can have attributes that provide additional information or modify their behavior. Attributes are specified within the opening tag of an element and consist of a name-value pair. For example, the "src" attribute in an image tag specifies the URL or path of the image file. Attributes can be used to define styles, specify links, set values, or add functionality to elements.
4. Nesting and Hierarchy: HTML elements can be nested within each other to create a hierarchical structure. This nesting determines the parent-child relationships between elements. For example, a paragraph element ("<p>") can contain a link element ("<a>"), which in turn can contain text or other elements. The hierarchy of elements contributes to the overall structure and organization of the web page.
5. Head and Body Sections: An HTML document is divided into two main sections: the head section and the body section. The head section contains meta-information about the web page, such as the title, character encoding, linked stylesheets, or scripts. The body section holds the visible content of the web page, including text, images, links, and other elements.
6. Document Structure: The overall structure of an HTML document follows a specific pattern. It begins with the DOCTYPE declaration, followed by the opening "<html>" tag. The "<html>" tag encloses the head and body sections. Inside the body section, the web page content is defined using various HTML elements and tags.
7. Semantic HTML: Semantic HTML refers to using HTML elements that carry meaningful and descriptive tags based on their purpose. It helps improve the accessibility, search engine optimization, and overall structure of a web page. Examples of semantic elements include "<header>", "<nav>", "<section>", "<article>", "<footer>", and more.

Understanding the HTML structure is essential for web scraping, as it allows you to locate and extract specific elements or content from web pages. By analyzing the HTML structure, you can identify the appropriate tags, classes, IDs, or attributes to target and retrieve the desired data during web scraping or other web development tasks.

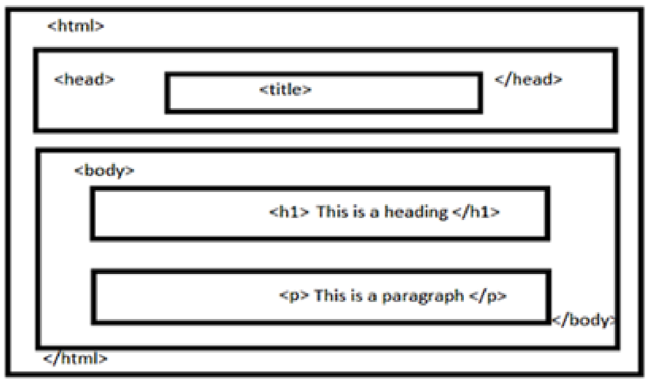

Figure 4. HTML Structure

IV. Conclusion

In conclusion, Open Source Intelligence (OSINT) combined with web scraping through APIs offers powerful capabilities for gathering and analyzing publicly available information from various online sources. By leveraging web APIs, developers can automate the process of extracting data from websites in a structured and efficient manner. Here are some key points to summarize the benefits and considerations of OSINT with web scraping through APIs:

Benefits:

1. Access to Rich and Diverse Data: Web scraping through APIs allows access to a wide range of data sources, including news sites, social media platforms, government databases, and more. This enables comprehensive data collection and analysis for OSINT purposes.
2. Structured Data Retrieval: APIs provide a standardized and structured way to retrieve data, typically in formats like JSON or XML. This allows for easier parsing, extraction, and processing of relevant information, saving time and effort compared to traditional web scraping methods.

3. Efficiency and Automation: By automating the data retrieval process through APIs, developers can save significant time and resources. API requests can be scheduled, run in batches, or triggered based on specific events, enabling efficient and continuous data gathering for OSINT applications.
4. Authentication and Authorization: Web APIs often incorporate robust authorization mechanisms, such as OAuth or API keys, which allow for secure access to protected resources. This ensures that data is accessed and retrieved by authorized parties, maintaining data integrity and privacy.
5. Integration and Scalability: OSINT applications can integrate data from multiple APIs, combining diverse information sources into a comprehensive analysis. APIs also provide scalability, as developers can scale their scraping processes to handle large volumes of data without overloading the target websites.

Considerations:

6. API Limitations and Terms of Service: APIs may have usage limitations, such as rate limiting or data access restrictions. It is essential to review and comply with the terms of service or usage policies of the API providers to ensure responsible and legal usage.
7. Data Quality and Verification: While APIs offer structured data, the quality and accuracy of the information depend on the source and the API implementation. Data retrieved through web scraping should be carefully validated and verified to ensure its reliability and integrity for OSINT analysis.
8. API Changes and Maintenance: APIs may undergo changes, version updates, or even be discontinued. Developers need to be aware of such changes and adapt their scraping processes accordingly to maintain the functionality and reliability of their OSINT applications.
9. Privacy and Ethical Considerations: When scraping data through APIs, it is crucial to respect the privacy and legal boundaries set by the website owners and API providers. Developers should adhere to ethical practices, comply with relevant laws and regulations, and prioritize user consent and data protection.

In summary, OSINT applications powered by web scraping through APIs provide valuable insights and intelligence from publicly available online data sources. By leveraging APIs' structured data retrieval capabilities, developers can automate the data gathering process, enabling efficient analysis and informed decision-making in various domains, including security, market research, sentiment analysis, and more. However, it is essential to understand and comply with the terms of service, ensure data quality, and maintain ethical practices to maximize the benefits and avoid any legal or ethical implications.

ACKNOWLEDGEMENT

This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS/CCCDI - UEFISCDI, project number PN-III-P2-2.1-SOL-2021-0063, within PNCDI III.

REFERENCES

[1] R. A. Furuhaug, "*Open Source Intelligence Methodology*", University College Dublin, 15 May 2019.
[2] N. A. Hassan, R. Hijazi, "Open Source Intelligence Methods and Tools: A Practical Guide to Online Intelligence", 2018.
[3] R. Ghioni, M. Taddeo and L. Floridi, "*Open source intelligence and AI: a systematic review of the GELSI literature*", Springer Nature, 28 Jan. 2023.
[4] G. T. Ungureanu, "*OPEN SOURCE INTELLIGENCE (OSINT). THE WAY AHEAD*", ”Mihai Viteazul” National Intelligence Academy, Bucharest, Romania, 2021
[5] B. Michael, "Urce Intelligence Techniques Resources For Searching And Analyzing Online Information", CCI Publishing, 2022.
[6] H. J. Williams, I. Blum, "Defining Second Generation Open Source Intelligence (OSINT) for the Defense Enterprise", RAND Corporation, 2023.
[7] R. A. Pinto, M. J. Hernández, C. C. Pinzón, D. O. Díaz and J. C. C. García, "Open source intelligence (OSINT) as support of cybersecurity operations. Use of OSINT in a colombian context and sentiment Analysis", Universitad Distrital “Francisco Jose de Caldas”, 03 Jun. 2018.
[8] https://www.imperva.com/learn/application-security/ open-source-intelligence-osint/ accessed 25th June 2023.
[9] https://www.sans.org/blog/what-is-open-source-intelligence/ accessed 22th June 2023.