From Raw Social Data to Actionable Intelligence: The SocHunter Approach

Stefano Bistarelli[1,†], Cristian Cerami[1,*,†] and Francesco Santini[1,†]

[1]Università degli Studi di Perugia, Perugia, Italy

Abstract

We present *SocHunter*, a tool designed to automate *Open Source Intelligence (OSINT)* operations by monitoring potential threats across *Reddit*, *YouTube*, and *Telegram*, with a specific focus on detecting messages that discuss *terrorist groups* or warn of possible *hacker attacks* against public organizations or companies. In this paper, we present the tool's architecture and operational logic, highlighting the key components that enable our objective. We discuss techniques for identifying scraping targets across all three social networks, with a specific focus on detecting major hacker and terrorist groups on Telegram. Finally, through three case studies, we demonstrate the effectiveness of SocHunter by presenting analysis results visualized in the Wazuh SIEM interface.

Keywords

OSINT, Cyber Threat Intelligence (CTI), Social Media Monitoring (SMM)

1. Introduction

Open Source Intelligence (*OSINT*) is an essential component of modern cybersecurity and threat-monitoring operations [1]. As online ecosystems expand across traditional social networks and emerging platforms, malicious actors increasingly exploit these channels to coordinate cyberattacks, spread extremist propaganda, and disseminate operational information [2, 3]. However, detecting such activity in real time remains a complex challenge: the vast volume of user-generated content, the diversity of languages and communication styles, and the prevalence of obfuscation techniques undermine the effectiveness of manual monitoring. Automated OSINT systems are therefore critical for enabling early threat identification, reducing analyst workload, and improving situational awareness. In recent years, these challenges have intensified due to the rapid proliferation of multilingual, irregular, and deliberately obfuscated content, combined with the instability of online communities, particularly those associated with cybercrime or extremism, which frequently migrate across platforms to evade bans. As a result, human analysts face an increasingly unmanageable information landscape in which relevant signals are deeply embedded in noisy, fast-moving data streams.

In this context, we present *SocHunter*, a tool that automates OSINT collection and analysis across *Reddit*, *YouTube*, and *Telegram*. The goal of SocHunter is to detect messages that refer to terrorist organizations or indicate potential cyberattacks against companies and public institutions. To accomplish this, the tool integrates scraping technologies with advanced *Natural Language Processing* (*NLP*) pipelines, multilingual translation, *Named Entity Recognition* (*NER*), fuzzy matching against user-provided blacklists, and a threat-scoring framework. Beyond simply aggregating data, SocHunter aims to bridge the gap between unstructured social-media chatter and operational intelligence needs, transforming raw messages into structured indicators that analysts can act on immediately.

SocHunter systematically transforms raw social-media messages into actionable intelligence by extracting key entities, evaluating their relevance to predefined risk categories, and assigning a "danger level", which is then forwarded to a *Wazuh SIEM* instance for alerting and visualization. A *Security Information and Event Management* [4] (*SIEM*) system is a platform that centralizes log data, correlates

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT

*Corresponding author.

†These authors contributed equally.

stefano.bistarelli@unipg.it (S. Bistarelli); cristian.cerami@studenti.unipg.it (C. Cerami); francesco.santini@unipg.it (F. Santini)

events, and generates alerts to support real-time threat detection and incident response; therefore, integrating SocHunter with Wazuh SIEM enables continuous monitoring and automated security alerting. This integration with the SIEM infrastructure enables automated, continuous monitoring workflows that would otherwise require extensive manual effort. Moreover, by storing both messages and metadata in MongoDB, SocHunter supports longitudinal analysis and incremental scraping, enabling persistent monitoring of evolving online ecosystems.

A central challenge addressed by SocHunter is the heterogeneous, often noisy nature of social-media text. Messages are frequently written in multiple languages, include stylistic or Unicode-based obfuscations, or reference organizations indirectly or with deliberate misspellings. To address this, SocHunter incorporates automatic language identification, English translation, entity extraction using SpaCy and Microsoft Presidio Analyzer, and similarity analysis via the TheFuzz library to detect approximate matches against user-defined blacklists. Hence, the tool captures relevant content even when entities appear in abbreviated, altered, or partially obfuscated forms, a common occurrence in both cybercriminal and extremist communities. SocHunter's modular architecture and customizable recognizer system let users tailor the tool to specific contexts, define new entity types, and extend detection logic without changing the source code, keeping the system effective as threat actors evolve their communication patterns.

Finally, the tool not only flags high-risk content but also stores all results and contextual metadata in a *MongoDB* database, enabling incremental scraping and facilitating longitudinal analysis. Notifications for WARNING and DANGER events are delivered to security analysts via *Discord* or email, ensuring real-time responsiveness. The result is an adaptable, extensible, and operationally practical system for automating OSINT activities on multiple platforms. Beyond its architectural design, SocHunter has been validated through three real-world use cases that demonstrate its effectiveness in detecting both cyber-threat activity and terrorism-related content:

Telegram (Hacking): By monitoring 436 chats and analyzing over $133,000$ messages, SocHunter successfully identified indicators of cyberattacks against major organizations, including DDoS operations, data leaks, and ransomware-related communications. In one notable case, the tool flagged a DDoS attack targeting a University in Italy, correctly scoring the event as DANGER and detecting obfuscated references that matched entries in the organization's blacklist.

Telegram (Terrorism): Focusing on channels related to the recent Israel-Palestinian conflict, the tool analyzed more than $55,000$ messages and reliably detected references to terrorist organizations such as the Al-Qassam Brigades[1]. SocHunter found a link to the official Al-Qassam Brigades website, enabling analysts to uncover their new Telegram channels that had remained hidden after earlier account bans.

Reddit (Terrorism): Analysis of 18 subreddits, comprising more than $3,800$ submissions and $116,000$ comments, revealed a high concentration of extremist-related content among the most active users. Some accounts exhibited a DANGER-level match rate of up to 70% of their activity, demonstrating that SocHunter can highlight high-risk users and communities at scale, an effort that would be prohibitively time-consuming through manual inspection.

These results demonstrate SocHunter's ability not only to detect individual high-risk messages but also to uncover broader behavioral patterns, discover hidden communication channels, and support intelligence workflows through structured high-fidelity analysis.

This paper presents the architecture, implementation, and evaluation of SocHunter. After introducing the core technologies used (Sect. 2), we describe the system's design and operational logic (Sect. 3), followed by three case studies that illustrate its effectiveness in real-world OSINT scenarios on Telegram and Reddit (Sect. 4). Finally, Sect. 5 draws conclusions and outlines possible future work.

[1] https://en.wikipedia.org/wiki/Al-Qassam_Brigades

2. Background

This section provides an overview of the main libraries used to create the SocHunter tool, ranging from scraping libraries (Asyncpraw, Google-Api-Python-Client, and Telethon) to text analysis libraries (SpaCy, Microsoft Presidio Analyzer, and TheFuzz).

As shown in Fig. 1, the first step of the tool is to scrape data from social networks. To scrape Reddit submissions and related comments, it is necessary to use Reddit's [5] Application Programming Interface (API). Since the project is implemented in Python, we used the open-source *Asyncpraw* [6] library. To scrape YouTube videos, it is necessary to use the *YouTube Data API (v3)* [7], officially provided by Google. Since the project is implemented in Python, we used the open-source *Google-Api-Python-Client* [8] library. For Telegram, several libraries facilitate the interaction with the User API [9], enabling the creation of automated applications known as *userbots*. Since the project is implemented in Python, we used the open-source *Telethon* library [10].

As the second step, the tool translates the extracted text if it is not already in English. This can be done using two libraries: *Translators* [11] and *Transformers* [12]. The first is a Python library for text translation that supports various translation engines. Among them, two have been implemented in the project due to their specific characteristics: *Google* and *MyMemory* [13]. The first supports 134 languages and, for most of them, produces good translations that maintain the original sentence's meaning. However, in a few languages, such as Hindi, the translations are distorted, losing the original sense of the sentence. Currently, based on how the Translators tool has been implemented, *there is no maximum limit on the number of characters that can be translated for free.* As observed in the Google class[2] of the Translators library, it accepts some input language codes formatted according to `ISO 639 (1`[8] `or 3`[3]`)` and `IETF BCP 47`[4] standards. MyMemory, instead, supports 330 languages and claims to be the world's largest *Translation Memory* (TM). While the library provides high-quality translations, its free tier limits usage to $5,000$ characters per day. As observed in the `MyMemory`[5] class of the Translators library, it accepts some input language codes formatted according to `ISO 639 (1`[8] `or 3`[3]`)` and `ISO 3166-1`[6] standards.

Currently, the project has also implemented the use of the *Transformers* library with the `facebook/nllb-200-3.3B`[7] [14] model, an open-source multilingual translation model developed by Meta AI [15] to translate 200 different languages. There are various models in the `NLLB-200` family that vary in size: `nllb-200-distilled-600M`, `nllb-200-distilled-1.3B`, `nllb-200-1.3B`, `nllb-200-3.3B`[7] and `nllb-moe-54b`. The one we chose strikes the right balance between translation quality and required resources, even though, due to limited hardware, it was tested only on the CPU (to ensure optimal performance, it requires at least a GPU with 16GB of VRAM). Despite suboptimal CPU performance, we retained this implementation because it enables GPU acceleration with minimal configuration changes and reduces reliance on external translation services, which often impose daily limits on the number of free characters.

To enable translators to operate correctly, the tool must first automatically determine the input language of the text. For this purpose, we use *LangID* [16], a Python library that can recognize 97 different languages, which are returned as output using their respective `ISO 639-1` language codes[8]. The recognition is fairly robust, working well even in the presence of HTML or XML markup. However, we observed that it is sometimes influenced by the "style" of the text, which is why it is advisable to first normalize it using functions[9] that convert stylized or special (Unicode) characters into a simpler, more uniform ASCII representation. An example could be the conversion of the string "***TEAM FEARLESS***", formed using the characters "Mathematical Sans-Serif Bold Italic" from the Unicode Mathematical

[2] https://github.com/UlionTse/translators/blob/90266137e9e30bf695776a75ca5bde12db2e4adf/translators/server.py#L374
[3] https://en.wikipedia.org/wiki/ISO_639-3
[4] https://en.wikipedia.org/wiki/IETF_language_tag
[5] https://github.com/UlionTse/translators/blob/90266137e9e30bf695776a75ca5bde12db2e4adf/translators/server.py#L4230
[6] https://en.wikipedia.org/wiki/ISO_3166-1
[7] https://huggingface.co/facebook/nllb-200-3.3B
[8] https://en.wikipedia.org/wiki/List_of_ISO_639_language_codes
[9] https://en.wikipedia.org/wiki/Unicode_equivalence

Table 1
Entities recognized by the `en_core_web_lg`[11] model.

PERSON	People, including fictional
NORP	Nationalities or religious or political groups
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
MONEY	Monetary values, including unit
TIME	Times smaller than a day
DATE	Absolute or relative dates or periods
EVENT	Named hurricanes, battles, wars, sports events, etc.
LANGUAGE	Any named language
LAW	Named documents made into laws.
PRODUCT	Objects, vehicles, foods, etc. (not services)
WORK_OF_ART	Titles of books, songs, etc.
PERCENT	Percentage, including "%"
QUANTITY	Measurements, as of weight or distance
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type

Table 2
Mapping between the entities detected by Presidio's `SpacyRecognizer` and the actual entities detected by SpaCy.[12]

ENTITY LABELS	DEFINITION	MAPPING ON SPACY
PERSON	People, including fictional	PERSON
NRP	Nationalities or religious or political groups	NORP
LOCATION	Countries, cities, states, mountain ranges, bodies of water, etc.	GPE, LOC
ORGANIZATION	Companies, agencies, institutions, etc.	ORG
DATE_TIME	Times smaller than a day and absolute or relative dates or periods	DATE, TIME

Alphanumeric Symbols set (`U+1D5E0 - U+1D7FF`), into "TEAM FEARLESS".

After translation, the text is analyzed using the libraries *SpaCy* [17] and *Microsoft Presidio Analyzer* [18]. The first is mainly used for the automatic recognition of key entities (see Tab. 1) within messages extracted from Reddit, Telegram, and YouTube, thanks to its *Named Entity Recognition* (NER) component. These entities, such as the names of people, companies, and criminal organizations, are then compared against lists of keywords deemed relevant, defined in blacklists, to identify content potentially relevant to our goal (messages mentioning specific terrorist groups or warnings of possible hacker attacks against public organizations or companies). The chosen model is `en_core_web_lg`[10], which enables analysis of English texts without training a custom model.

The Microsoft Presidio Analyzer has been configured to use the SpaCy `NlpEngineProvider` and the `en_core_web_lg`[10] model, resulting in the creation of the `SpacyRecognizer` and thus the identification of the entities shown in Tab. 2. In addition to this recognizer and the default ones[13], we developed custom recognizers to detect entities related to hacker attacks, such as IP addresses, crypto wallets, names of known vulnerabilities (CVEs), keywords associated with DDoS attacks, data leaks,

[10] https://spacy.io/models/en/#en_core_web_lg
[11] See the "Label Scheme", "NER" section at: https://spacy.io/models/en#en_core_web_lg
[12] https://github.com/microsoft/presidio/blob/611f51dfb1642af705853035efc9f4ed39f4a718/presidio-analyzer/presidio_analyzer/predefined_recognizers/nlp_engine_recognizers/spacy_recognizer.py
[13] https://microsoft.github.io/presidio/supported_entities/

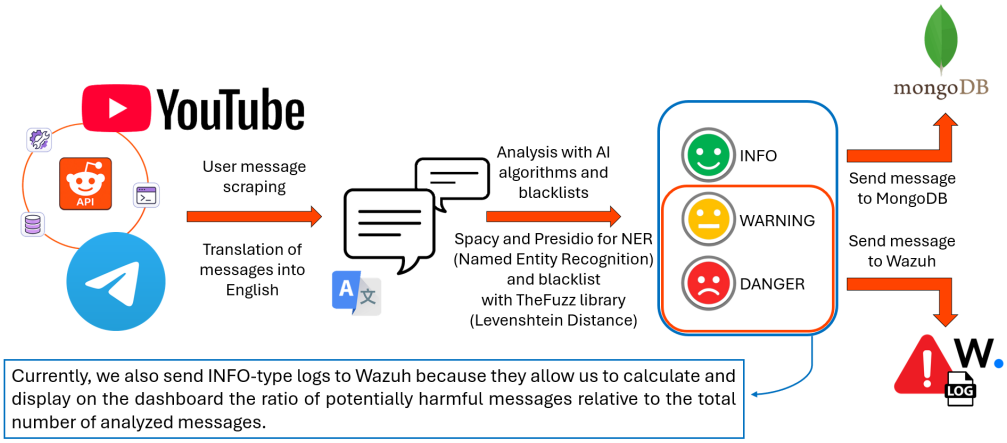

Figure 1: Simplified functioning of SocHunter.

ransomware, etc. Furthermore, thanks to the modular architecture of Presidio, it has allowed us to implement a logic enabling end users to define new custom entities via a simple JSON file (see Sect. 3.5), which describes the desired regex patterns and any contextual words that can increase the recognition confidence scores for the corresponding entity. This enables rapid system customization without requiring direct modifications to source code, thereby enhancing the tool's flexibility and adaptability across various operational scenarios.

Some of the identified entities are searched within blacklists provided by the user to verify possible matches, which may not be exact but also partial. For this purpose, we use *TheFuzz* [19], an open-source library written in Python that allows the comparison of strings to calculate their similarity. We used it because when working with unstructured data, such as the messages extracted from the three previously mentioned social networks, the entities to be recognized may appear with slight variations compared to those present in the blacklists. Therefore, the implemented Levenshtein distance algorithm enables us to overcome the limitations of traditional text comparison based on exact matches, thereby identifying even partial or approximate correspondences. Its use is thus necessary for searching for possible matches between entities because, often, company or organization names appear in abbreviated form, with spelling errors, or even with characters deliberately replaced to avoid automatic recognition. To achieve this, as explained in Sect. 3.3, we evaluate combinations of scores produced by *WRatio* and *token_set_ratio* scorers.

3. SocHunter

The following sections describe SocHunter's architecture, components, functionality, and configuration.

3.1. System Functioning

Figure 1 visually describes how the tool works. The process begins by scraping messages from social networks using user-defined configuration files (`.CSV`), one per social platform. Since we use the `en_core_web_lg` model in both SpaCy and Microsoft Presidio Analyzer (both trained for English-language analysis), all messages must first be translated into English. Subsequently, we perform the analysis, starting with SpaCy, which detects entities (such as people, organizations, political, religious, or terrorist groups) in the messages and checks for fuzzy matches (so, not necessarily exact) within the blacklists provided by the user as input.

If necessary, based on certain conditions, such as the presence of a specific company blacklist in the configuration, we then proceed to use Presidio, which searches for entities related to hacker attacks, such as IP addresses, crypto wallets, known vulnerability names (CVEs), keywords related to DDoS, data leaks, ransomware, and more. According to certain criteria, which will be discussed in more detail

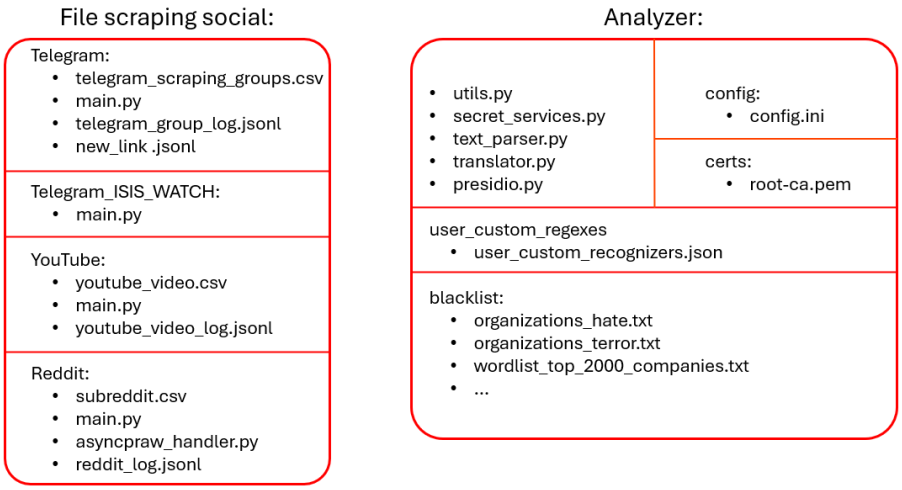

Figure 2: The SocHunter architecture.

in Sect. 3.4, the tool assigns a "`danger_level`" to the message, which can be "`INFO`", "`WARNING`", or "`DANGER`", with `INFO < WARNING < DANGER`. The analysis results are then sent to Wazuh SIEM, while the same results, together with all related scraping context data, are sent to the MongoDB database [20], which is indispensable for accurately analyzing only the newly read messages. For messages with `danger_level` `WARNING` or `DANGER`, Wazuh notifies the user via a dedicated Discord webhook [21] or, if configured, by email. Discord notifications are also sent if the program terminates unexpectedly.

3.2. Architecture

The tool architecture, as shown in Fig. 2, can be logically divided into two main components: "File scraping social" and "Analyzer".

3.2.1. File scraping social

Inside the "File scraping social" component, as the name suggests, we include the directories of the social media platforms from which we intend to scrape and analyze messages: Reddit, YouTube, and Telegram. Each directory contains a standalone program, and all follow a similar structure, with the most important files being "`main.py`" and "`.csv`" files. The first is a Python file (for Reddit, there is also "`asyncpraw_handler.py`") that contains the corresponding scraping logic, using the libraries discussed in Sect. 2. The "`.csv`" file contains the scraping configuration for an individual subreddit, video, group, or channel, allowing the analysis of each data source to be as customizable as possible.

For Telegram, in addition to scraping the "classic" chats, we also analyzed messages from `ISIS WATCH` [22], an official Telegram channel that reports the daily number of terrorist groups, channels, and bots banned the previous day.

3.2.2. Analyzer

Within the "analyzer" component, we have all the files required to analyze the messages we receive and determine whether they discuss a specific criminal/terrorist organization or a potential hacker attack. This component is common to all defined scraping programs and contains the following files: utils.py, secret_services.py, translator.py, presidio.py, and text_parser.py. The first contains general utility functions such as initialization of SpaCy and PyMongo [23, 24] (MongoDB) libraries, initialization of notification/alert mechanisms for Discord and Wazuh, reading and validation of blacklists, cleaning read messages, sending logs to MongoDB, and safe termination of the program in case of errors. `secret_services.py` handles the reading of the main tool configuration file (`config.ini`) and the initialization of the social clients. `translator.py` manages translators and message translation by

mapping language codes returned by LangID to those accepted as input by the translators defined in the Translators and Transformers libraries (see Sect. 2). `presidio.py` manages the handling, creation, and use of recognizers from Microsoft Presidio Analyzer (see Sect. 2) for the identification of entities within messages. Specifically, it creates a PresidioAnalyzer object containing two Analyzer instances: `company_analyzer` and `general_analyzer`. The first one uses the custom recognizers defined by the user in the "`user_custom_recognizers.json`" file, "`for_companies`" section (see Sect. 3.5), the default[13] Presidio recognizers, and all those defined by us to detect indicators of potential cyber-attacks such as sensitive entities (IP addresses, phone numbers, credit cards, crypto wallets, emails, passwords, URLs) and keywords associated with DDoS attacks, data leaks, ransomware, defacement, initial access, and vulnerabilities (CVE). This analyzer is used only with the "companies blacklist" (see Sect. 3.5). When a recognizer matches one of its regexes, the identified entity is reported with the corresponding label and taken as-is, without being compared against the blacklist values. The `general_analyzer`, on the other hand, uses the custom recognizers defined by the user in the "`user_custom_recognizers.json`" file, "`for_all_blacklists`" section (see Sect. 3.5). It does not load the predefined[13] Presidio recognizers, but its predictions are merged with those of SpaCy and used to identify entities to search for in the blacklists. Lastly, `text_parser.py` handles message analysis and related tasks, including text preprocessing and translation, entity recognition (SpaCy + Presidio), fuzzy matching against blacklists, calculating the `danger_level`, updating MongoDB documents, and sending events to Wazuh. During text preprocessing, several actions are performed to improve entity detection. These include removing markdown formatting, extracting file/photo/video names from Telegram messages, and extracting URL domains so that SpaCy can detect them as company or organization names.

3.3. Fuzzy Matching on Blacklist

To find the best match on the blacklist for an entity just identified by SpaCy or by the `general_analyzer`, various scorers from the TheFuzz library are used. The search returns both the textual value and two similarity scores (`fuzzy_value` for the `WRatio` scorer and `fuzzy_value_token_set` for the `token_set_ratio` scorer), which are then used to check if there is an actual match between the two strings. Three different cases are tested: full match, partial match, and match in obfuscated text. In the first case, it is checked whether either score equals `100/100`. In the second case, we tried to minimize false positives and false negatives by creating two different conditions for accepting the match, respectively: if the `fuzzy_value` is greater than or equal to 80, with a length difference between the two strings not exceeding two characters and with a `fuzzy_value_token_set` greater than or equal to 70, or if the `fuzzy_value_token_set` is greater than or equal to 90, so as to accept names, usually company names, that are often abbreviated (example: `"Microsoft Presidio An." -VS- "Microsoft Presidio Analyzer"`). In the third case, if the previous checks fail, we try to see if the identified entity has been slightly obfuscated with numbers or special characters (`@$!£€`). If so, we attempt to reconstruct it using a function that contains a simple mapping of visually similar characters, such as: `"1":"i", "@":"a","$":"s"` (example: `"H3zb0llah" => "Hezbollah"`). Once the "decoded" string is obtained, the full- and partial-match checks are retested.

3.4. Calculation of the Danger Level

As previously seen in Sect. 3.1, the tool assigns a "`danger_level`" to each analyzed message, which can be "`INFO`", "`WARNING`", or "`DANGER`" with `INFO < WARNING < DANGER`. There are two functions to calculate the `danger_level`, namely `danger_calculator()` and `danger_calculator_for_companies()`; the first is used for every message, while the second is used only if the blacklist of companies (among those to analyze) is present.

The function `danger_calculator()` returns a `danger_level` value of `DANGER` if at least one `fuzzy_value` or `fuzzy_value_token_set` (see Sect. 3.3) equals `100/100`, or if it has obtained an

```
-----------------------------------------------------
| SPACY | SPACY | SPACY | SPACY | SPACY | SPACY | SPACY |
-----------------------------------------------------
  NAME    -> 'PERSON', 'NORP', 'ORG', 'FAC', 'USER_CUSTOM_XXX' ('for_all_blacklists' section)
  DATE    -> 'DATE'
  TIME    -> 'TIME'
-----------------------------------------------------
| PRESIDIO | PRESIDIO | PRESIDIO | PRESIDIO | PRESIDIO |
-----------------------------------------------------
  IP_ADDRESS      -> 'IP_ADDRESS', 'IPV4', 'IPV6'
  PHONE_NUM       -> 'PHONE_NUMBER', 'PHONE_NUMBER_CUSTOM
  ID_NUM          -> 'US_SSN', 'US_PASSPORT', 'US_DRIVER_LICENSE'
  MONEY           -> 'MONEY'
  MONEY_WALLET    -> 'CRYPTO', 'CRYPTO_CUSTOM', 'CREDIT_CARD', 'CREDIT_CARD_CUSTOM', 'IBAN_CODE'
  EMAIL           -> 'EMAIL_ADDRESS', 'EMAIL_CUSTOM'
  PASSWORD        -> 'PASSWORD'
  URL             -> 'URL', 'URL_CUSTOM'
  DATA_SIZE       -> 'DATA_SIZE'
  CYBER_ATTACK    -> 'DDOS_ATTACK', 'DATA_LEAK_ATTACK', 'RANSOMWARE_ATTACK', 'INITIAL_ACCESS_ATTACK',
                     'VULNERABILITY_ATTACK', 'DEFACEMENT_ATTACK', 'GENERAL_ATTACK', 'ALERT'
  USER_CUSTOM_XXX -> 'USER_CUSTOM_XXX' ('for_companies' section)
```

Figure 3: Entity labels (to the left of the arrow "`->`") accepted in `user_combinations`. To the right of the arrow are the names of individual entities recognized by SpaCy or by the Presidio recognizers.

`average_score >= 90` (where `average_score` represents the average similarity score given by the sum of individual scores divided by the total number of entities for which a match was accepted within the blacklist). Thus, a `danger_level` of DANGER indicates that the analyzed message indeed refers to something connected to the input blacklists. The function returns WARNING if it calculates that `average_score < 90`, indicating that the analyzed message is likely related to the input blacklists but may be a false positive. The function returns INFO if no match was accepted within the blacklists, indicating that the message does not contain entities linking it to the input blacklists.

The function `danger_calculator_for_companies()` returns a `danger_level` based on two input values: `entities_found` and `user_combinations`. The first represents the list of labels of the entities identified by SpaCy, `general_analyzer`, and `company_analyzer` (see Fig. 3 for the full list of possible labels). The second is a list of combinations in the following format:

`ENTITY_1+...+ENTITY_N->DANGER_LEVEL | ENTITY_1+...+ENTITY_M->DANGER_LEVEL`

so, combinations of entity labels, separated by the character "`|`", to which the user assigns a specific `danger_level` as part of the scraping configuration for each individual subreddit, video, group, or channel. The entity labels that can be used are those to the left of the arrow "`->`" in Fig. 3. As shown, it is also possible to use entities identified by the user's custom recognizers in the `user_custom_recognizers.json` file, under the `for_companies` section. Those created in the `for_all_blacklists` section are mapped in SpaCy with the label NAME, as they are intended to find matches within the blacklists.

Once the `danger_level` has been calculated for all provided blacklists, the maximum value is identified and used as the final `danger_level` for the analyzed message.

3.5. Tool Configuration

The project comprises several configuration files: one main file that configures general settings across all social platforms, and others, as shown in Sect. 3.2.1, that specify the scraping settings for the individual Telegram chats, YouTube videos, and Reddit subreddits to be analyzed.

The main file is divided into sections, each identifiable by a title enclosed in square brackets. Each section has its own function, including defining credentials for creating social clients or for using the Wazuh API, configuring Discord webhooks and the translator type to use, specifying the company blacklist name, and many other settings that, especially for Telegram, modify the scraping logic. For the latter, and to ensure the tool operates automatically correctly, users can define automatic join modes for new chats whose links are found in analyzed messages. This is configured by setting join filters that

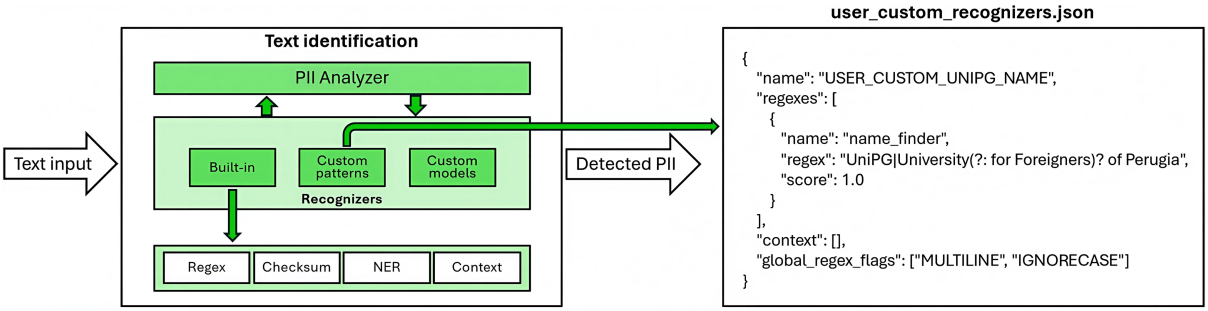

Figure 4: On the left, the functioning of Microsoft Presidio Analyzer. On the right, an example of a user-defined custom recognizer (see Sect. 3.5).

operate based on the type of chat detected (*group* or *channel*) and the "type" of message itself (whether it is a *backup*-for a banned chat, *alliance*-usually with other hacker groups, *forward*-for forwarded messages, or "*link*"-meaning neither backup, alliance, nor forward).

The `.csv` configuration file for Telegram must contain at least 4 elements for each row to comply with the required structure: chat ID, number of messages to analyze during the initial scraping (starting from the most recent; in subsequent runs, all new messages will be analyzed), `user_combinations` (as described in Sect. 3.4), and the list of blacklists to be used.

The `.csv` configuration file for Reddit must contain at least 6 elements for each row to comply with the required structure: subreddit name, three values representing the number of submissions to retrieve using *new*, *top*, and *hot* sorting orders, `user_combinations` (as described in Sect. 3.4), and the list of blacklists to be used.

The `.csv` configuration file for YouTube must contain at least 3 elements for each row to comply with the required structure: video ID, `user_combinations` (as described in Sect. 3.4), and the list of blacklists to be used.

Finally, a `.json` file allows the user to create custom recognizers (see Fig. 4) that will be integrated into Presidio `general_analyzer` and `company_analyzer`. The file consists of two dictionaries, `for_all_blacklist` and `for_companies`; the first is to create recognizers to add to the `general_analyzer` and thus to search for matches within the blacklists, while the second is to create recognizers to add to the `company_analyzer` and therefore to search for entities whose labels can be used in `user_combinations`.

In addition to these configuration files, there is a directory named "blacklist" that contains all possible blacklists used during message analysis. Among these, only one is defined as the "company blacklist" in the main configuration file, which enables the detection of hacking attacks. See Sect. 4 for a definition of the main blacklists used by SocHunter.

3.6. Identification of Scraping Targets

Before explaining the methods we used to identify data sources for scraping, it is essential to note that we primarily use Telegram and, to a lesser extent, Reddit for cyberattack-related data. To search for messages discussing specific terrorist groups, we use all three social media platforms. Furthermore, in terms of terrorist content, we have decided to focus mainly on the ongoing Palestinian conflict.

For Telegram and the detection of cyber attacks, we relied on the scientific literature [25, 26], GitHub repositories (such as deepdarkCTI [27] and Awesome Telegram OSINT [28]), and specialized websites (such as Breachsense [29]). Through these sources, we identified numerous channels, mainly associated with hacker groups (among the most famous: NoName057(16), DieNet, Keymous+, Dark Storm Team, Scattered Lapsus Hunters) as well as Cyber Threat News outlets, including Dark Web Informer, RansomfeedNews, The Hacker News, and Hackmanac Cyber News. If previously collected links to hacker channels were invalid, we searched for channel names using Telegago [30], a *Google Custom Search Engine* (*CSE*) for OSINT on public Telegram content; if still unsuccessful, we consulted

third-party catalogs of active public Telegram channels, such as Telemetr.io [31] and TGStat.ru [32].

The same research methodology was applied to the search for channels dealing with terrorist group content; we relied on the scientific literature [33] and specialized websites such as the Counter Extremism Project (CEP) [34] and the Institute for National Security Studies (INSS) [35]. Through these sources, we identified numerous channels, mainly news outlets (such as Palestine Updates, Enemy Watch, The Frontline, AlHaq News, Israel Today, Quds News Network, and Warfare Analysis NEWS) as well as some terrorist channels, such as ISNAD-Palestine Movement, طوفان الأمة (@TofanOmah25 - Hamas Al-Aqsa Flood), المراسل العسكري Military Correspondent Gaza (@Correspondent_Gaza - Al-Qassam Military Media Unit), كتائب القسام – الإعلام العسكري (@alqessam1) and كتائب الشهيد عز الدين القسام (@qs-sambrigades1) Al-Qassam brigades official channels, @AbuObaidahEnglish (Abu Ubaida channel - military spokesperson of Hamas's armed wing, the Al-Qassam Brigades).

For Reddit, we relied on the scientific literature [36, 33] to identify subreddits related to terrorism and the Palestinian war, while we focused more on specialized websites (such as Cyberpress [37]) to identify those related to cybersecurity. Among the identified subreddits, the most relevant proved to be those connected to the armed conflict (such as r/Palestine, r/IsraelPalestine, r/Israel_Palestine, r/terrorism, r/war, r/IsraelCrimes, ...), whereas those intended for detecting hacker attacks (such as r/databreach, r/Malware, r/ameeba, r/netsec, r/threatintel, r/InfoSecNews) proved to be of limited utility due to their low message publication rates.

For YouTube, we searched for recent videos discussing the Palestinian conflict and, in particular, specific terrorist groups of interest. To do so, we used tools such as the "youtube-search-tool" [38] from Aware Online and "youtube-geofind" [39], a web-based tool that allows searching for geographically tagged videos on YouTube by channel, topic, or location.

4. Case Studies

In the following section, we present three case studies of the tool, focusing on the social networks Telegram and Reddit. Some information in the screenshots has been obfuscated or redacted due to its sensitive nature, including the names of people involved in the attack and their contact information. All reported results remain accurate and unaffected by these obfuscations.

4.1. Telegram - Hacking

Between June and November 2025, we analyzed a total of 436 chats, comprising 10 unique news channels focused on hacker attacks and vulnerability reports (CVE), 290 unique hacker groups or channels related to the hacking world, 25 unique channels dedicated to the selling of services (such as VPS, databases, and DDoS attacks), and 3 unique hacker forums. The numbers above should be interpreted as unique entries, as hacker groups often maintain multiple chats due to frequent account bans. In total, we analyzed 133, 914 messages.

Usually, only one blacklist is used for analysis: "wordlist_top_2000_aziende.txt". As the name suggests, it lists the 2000 most profitable companies worldwide, sourced from Forbes [40]. More entries related to Italy have been added to this blacklist, such as the names of banks, universities, and government bodies. Since we configured this blacklist as the company blacklist (see Sect. 3.5), it was necessary to define the `user_combinations` for the individual chats to be analyzed; generally, these are:

NAME->WARNING | NAME+CYBER_ATTACK->DANGER | NAME+URL+PASSWORD->DANGER | NAME+MONEY ->DANGER | NAME+EMAIL->DANGER | NAME+MONEY_WALLET->DANGER | NAME+DATA_SIZE->DANGER | USER_CUSTOM_UNIPG_EMAIL->DANGER | USER_CUSTOM_UNIPG_URL->DANGER

We choose these combinations because, in the first case (NAME->WARNING), if a name on the blacklist appears without an accompanying attack indicator, it is prudent to inform the user with a WARNING. However, combinations such as NAME+URL+PASSWORD and NAME+DATA_SIZE trigger a DANGER alert, as they often reveal data leaks. Similarly, for NAME+EMAIL, NAME+MONEY, and NAME+MONEY_WALLET, we issued a DANGER alert because they could be associated with ransom demands after ransomware

attacks. To enhance protection for the University of Perugia, we configured some custom recognizers (USER_CUSTOM_UNIPG_NAME and USER_CUSTOM_PERUGIA) in the `for_all_blacklists` section of the `user_custom_recognizers.json` file (see Sect. 3.5). These recognizers, along with the last two combinations, detect attacks on this institution and generate DANGER-level alerts.

Among the numerous attacks detected by the tool against Italy, Fig. 5 shows the analysis of a message that contains a DDoS attack targeting an Italian university. The assigned `danger_level` is DANGER because the `data.occurrencies_for_companies` field contains the labels NAME, CYBER_ATTACK and URL, which satisfy the combination NAME+CYBER_ATTACK->DANGER. Examining the `data.occurrences` field, we can observe that the TheFuzz library performed effectively, assigning the maximum score to the detected entity despite it containing characters not present in the corresponding entity on the blacklist.

Figure 6 illustrates the network graph dynamically generated within Wazuh for the day the case study message was detected, filtered to display only connections to the group that published the attack. These connections result from interactions between the specific group (Red Eye Of Palestine) and any other Telegram chat referenced in the analyzed messages. SocHunter checks for usernames, invite links, or forwarded posts; upon detecting such references, it creates an edge between the corresponding nodes. Depending on the detection context and therefore on the type of message identified (see Sect. 3.5), the edge is labeled as backup, alliance, forward, or link.

The graph thereby enables clear identification of both the direct connection between Red Eye Of Palestine and the Red Wolf Cyber group (the attack's original author), from which the content was forwarded, and the role of Red Eye Of Palestine as a highly connected node. Indeed, as documented in several online reports [41, 42] and as visible in Fig. 7, that week the group led an offensive campaign against Israel, sharing content from numerous hacker and hacktivist groups.

So, integrating automated content analysis with relational visualization enables analysts to rapidly identify coordination structures and propagation patterns within offensive campaigns.

4.2. Telegram - Terrorism

As anticipated in Sect. 3.6, our analysis focused primarily on chats discussing the Palestinian conflict. Between June and November 2025, we analyzed a total of 25 chats, comprising 6 terrorist channels, 17

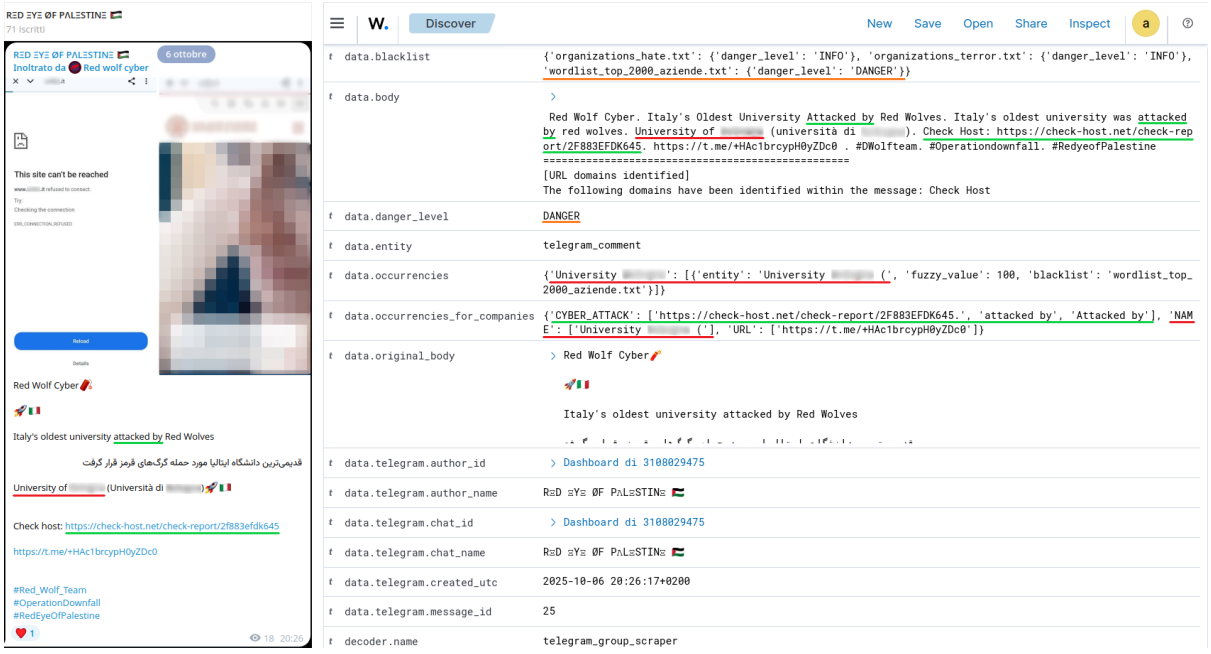

Figure 5: A partial screenshot of the Wazuh interface showing the analysis of the Telegram message, visible on the left, containing a DDoS attack against an Italian university.

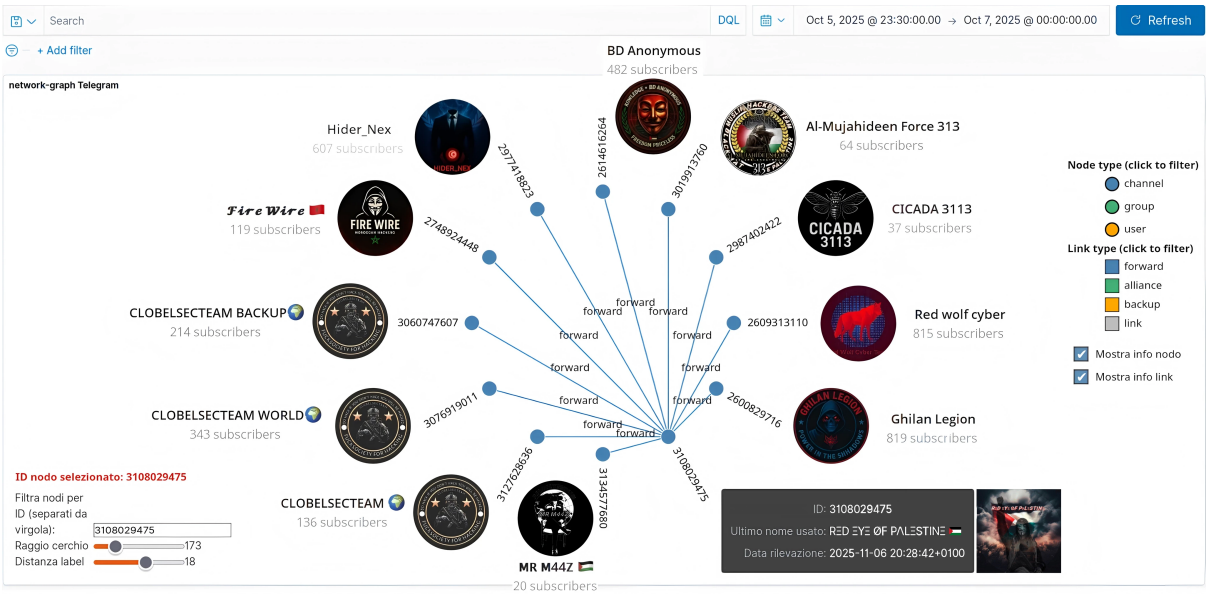

Figure 6: The network graph of Telegram chats linked to the hacktivist group Red Eye Of Palestine.

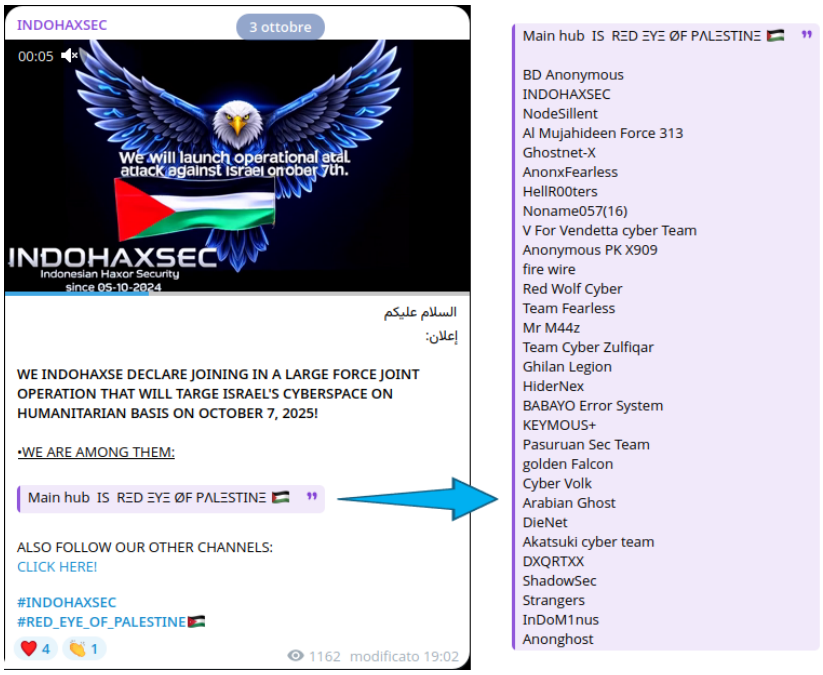

Figure 7: The Red Eye of Palestine is the main hub for a cyberattack coalition against Israel.

war-focused news channels, and 6 channels reporting more general news, yielding 55, 574 messages in total. Not all chats were analyzed simultaneously; some were manually added by the user, and others were automatically added by the tool upon discovery.

Three blacklists were used for analysis: “organizations_hate.txt”, “organizations_terror.txt”, and “individuals_terror.txt”. As their names suggest, these respectively contain hate groups, terrorist organizations (such as ISIS, Al-Qaeda, and Al-Qassam Brigades), and terrorist individuals (such as Osama bin Laden). These blacklists are primarily based on the leaked *DIO list* [43, 44] (Dangerous Individuals and Organizations), which Meta uses to limit unwanted content on its platforms, and are supplemented with additional Hamas-affiliated terrorist groups and leaders.

A particularly notable case detected by the tool, before we gained access to official terrorist group

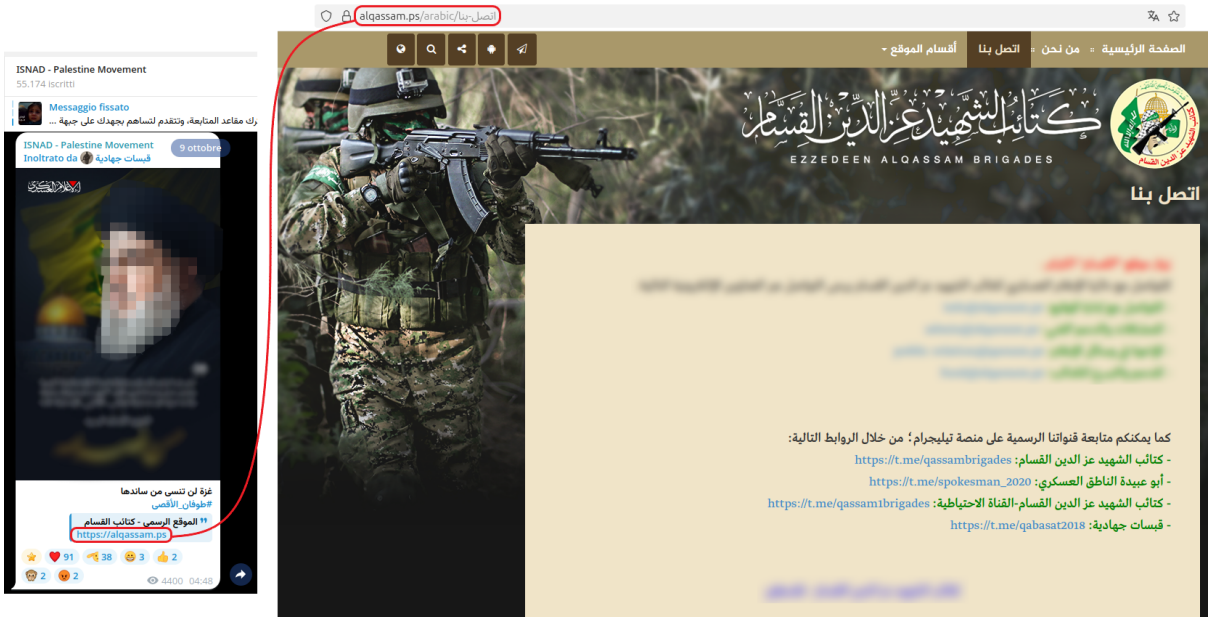

Figure 8: On the left, the Telegram message that led to the discovery of the Al-Qassam Brigades' official website, visible on the right, including the corresponding contacts section.

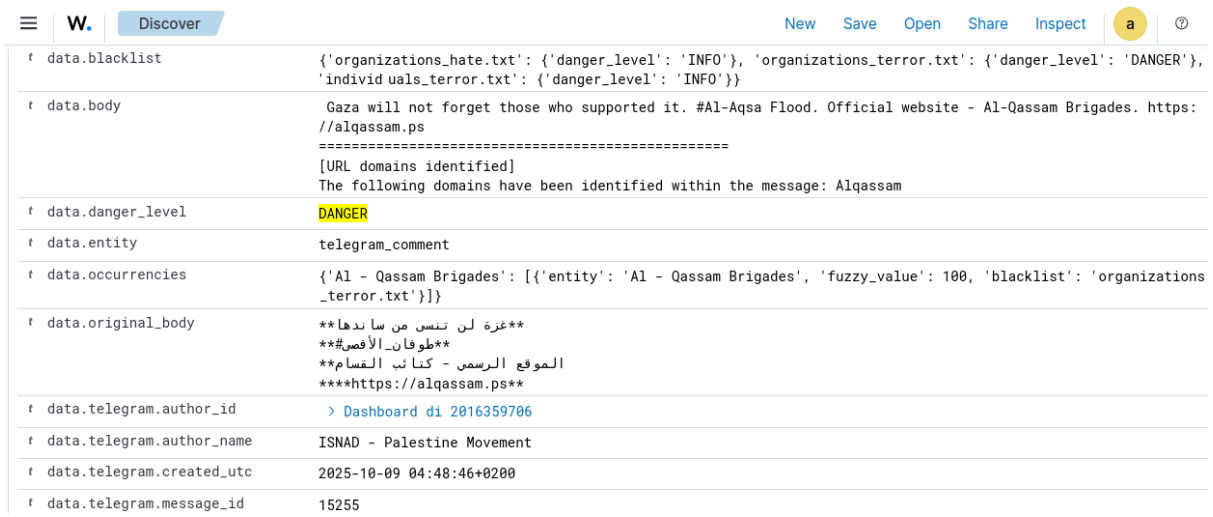

Figure 9: A partial Wazuh interface screenshot analyzing the Telegram message shown in Fig. 8.

channels, occurred during the scraping of ISNAD-Palestine Movement (@IsnadPalestin). The published message is shown in Fig. 8, and, as shown in Fig. 9, which displays the tool's analysis results within Wazuh, the user received an alert for a message with `danger_level` DANGER. Through text translation, we found a match for the entity "Al-Qassam Brigades", which led to a link to the terrorist group's official website. Within this site, as shown in Fig. 8, we found a contact section listing links to official Telegram channels; however, all had been banned, preventing direct access. Subsequently, using the tools described in Sect. 3.6, we searched for channel names and successfully identified the new profiles (@alqessam1, @qssambrigades1), thus expanding our data sources.

4.3. Reddit - Terrorism

Between October and mid-November 2025, we analyzed 18 subreddits, totaling 3,831 submissions and 116,586 comments; of these, as much as 86% came from only r/Palestine and r/IsraelPalestine. The same blacklists used in Sect. 4.1 and Sect. 4.2 were applied for the analysis. Reviewing the results

of the tool (see Tab. 3), we observed that among the most active users in the Palestinian subreddits (top 30 by number of posts and comments), there is a high rate of messages with `danger_level` DANGER, sometimes with an INFO/DANGER ratio as low as 29%. Furthermore, although many of these users primarily interact on r/IsraelPalestine, we found users who interacted on up to 4 of the analyzed subreddits. Manual detection of this information by an analyst would have taken days; thanks to SocHunter, the process could be automated, improving overall efficiency while maintaining excellent results through fuzzy matching.

Table 3
`danger_level` (posts and comments) by "terrorism" subreddits for top 30 users.

Rank 1) User: T**n**

Subreddit	INFO	WARN	DANGER	Tot
r/IsraelPalestine	556	0	285	841
r/Israel_Palestine	19	0	11	30

Rank 29) User: P**3**

Subreddit	INFO	WARN	DANGER	Tot
r/IsraelPalestine	73	0	249	322

Rank 14) User: w**9**

Subreddit	INFO	WARN	DANGER	Tot
r/war	4	0	2	6
r/Palestine	2	0	1	3
r/IsraelPalestine	269	0	205	474
r/Israel_Palestine	1	0	0	1

Rank 15) User: s**a**

Subreddit	INFO	WARN	DANGER	Tot
r/war	258	2	101	361
r/Palestine	6	0	2	8
r/IsraelPalestine	46	0	50	96

5. Conclusion

SocHunter, which operates on publicly available data for defensive security monitoring by authorized organizational teams under governance controls, demonstrates the feasibility and effectiveness of automating OSINT workflows across heterogeneous social media platforms. By integrating large-scale scraping, multilingual translation, entity recognition through SpaCy and Microsoft Presidio, and flexible fuzzy-matching logic, the system transforms unstructured online content into structured, actionable intelligence. The results from the three use cases confirm the tool's ability to detect indicators of cyberattacks, surface terrorism-related content, and reveal patterns of high-risk activity that are impractical to identify manually. In the presented case studies, SocHunter reliably identified threats ranging from DDoS operations and data leaks to references to extremist organizations, even in the presence of obfuscation, multilingual text, or irregular formatting.

Although primarily qualitative, these case studies complement a first quantitative validation. We used two *EuRepoC* cyber-incident datasets [45] that contain *only* news reports of attacks, rather than hacker group claims (our primary regex target). By adding a few new regexes, we achieved 97.8% of 2,957 messages (59.2% WARNING, 38.5% DANGER) and 97.9% of 4,243 (54.8% WARNING, 43.1% DANGER) flagged as threats. Since these threat-only datasets preclude precision calculation (no benign samples for false positives), the results demonstrate high recall and pipeline adaptability, despite the format differing from Telegram hacker chatter.

These evaluations validate the system's technical viability and highlight how automated tools such as SocHunter can significantly expand analysts' capabilities by reducing cognitive load, accelerating response times, and enabling signal correlation across platforms.

Beyond its operational performance, the work presented here also confirms the potential of automated OSINT systems to support analysts in discovering hidden communication channels, mapping patterns of coordinated behavior, and maintaining continuous monitoring of evolving online ecosystems. The architecture's flexibility (in particular, its modular recognizer design, customizable blacklist system, and integration with Wazuh SIEM) positions SocHunter as a practical and extensible foundation for real-world intelligence workflows.

A particularly promising direction for future work is the deeper integration of social media intelligence with traditional SIEM data. Traditional SIEM platforms typically focus on logs, alerts, and telemetry collected from hosts, networks, and security appliances; these sources provide high-fidelity technical signals but lack visibility into the human, organizational, and narrative dimensions that emerge on social media. By enriching SIEM workflows with OSINT-derived indicators, such as mentions of specific companies, announcements of upcoming attacks, or coordinated extremist messaging, analysts gain a broader situational picture.

As a further development of the system, the fuzzy-matching subsystem offers opportunities for refinement. Extending the obfuscation-handling logic by incorporating adaptive character-mapping tables learned from observed patterns (such as those used by hacker groups to evade detection) would improve recall without sacrificing precision.

Finally, introducing temporal and relational analytics, such as tracking the evolution of entities over time, computing inter-group similarities, or identifying synchronized posting behavior, would enable the system to progress from message-level alerts to richer situational intelligence.

Acknowledgments

S. Bistarelli and F. Santini are members of the INdAM Research group GNCS and members of Consorzio CINI. This work has been partially supported by: MUR PNRR project SERICS (PE00000014 AQuSDIT: CUP_H73C22000880001, COVERT: CUP_J93C23002310006), funded by the European Union – Next Generation EU; EU MUR PNRR project VITALITY (J97G22000170005), funded by the European Union – Next Generation EU; University of Perugia - Fondo Ricerca di Ateneo (2020, 2022) – Projects BLOCKCHAIN4FOODCHAIN, FICO, RATIONALISTS, "Civil Safety and Security for Society".

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT, Sonar, and Grammarly to check grammar and spelling, paraphrase, and reword. In addition, the authors used ChatGPT to generate Figure 1. After using these tools/services, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] A. Yadav, A. Kumar, V. Singh, Open-source intelligence: a comprehensive review of the current state, applications and future perspectives in cyber security, Artificial Intelligence Review 56 (2023) 12407–12438.

[2] U. Soler, S. Lovi, The role of telegram in coordinating cyber attacks and propaganda campaigns by russian hackers, Cybersecurity and Law 2 (2025) 316–334.

[3] FalconFeeds, How telegram is powering the new age of ddos, https://falconfeeds.io/blogs/telegram-ddos-defacement-hacker-claims, 2025.

[4] J. M. López Velásquez, S. M. Martínez Monterrubio, L. E. Sánchez Crespo, D. Garcia Rosado, Systematic review of siem technology: Siem-sc birth, International Journal of Information Security 22 (2023) 691–711.

[5] Reddit, Reddit documentation, https://support.reddithelp.com/hc/en-us, 2025.

[6] praw-dev community, asyncpraw: Asynchronous python reddit api wrapper, https://github.com/praw-dev/asyncpraw, 2019.

[7] G. Developers, Youtube data api v3 documentation, https://developers.google.com/youtube/v3, 2025.

[8] G. A. team, google-api-python-client: The official python client library for google's discovery based apis, https://github.com/googleapis/google-api-python-client, 2014.

[9] Telegram, Telegram documentation, https://telegram.org/faq, 2025.

[10] L. Exo, Lonamiwebs/telethon, https://github.com/LonamiWebs/Telethon, 2016.
[11] UlionTse, Translators: A library that aims to bring free, multiple, enjoyable translations to individuals and students in Python, https://github.com/UlionTse/translators, 2017.
[12] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.
[13] MyMemory, A collaborative language resource, https://mymemory.translated.net/doc/, 2025.
[14] H. Face, facebook/nllb-200-3.3b, https://huggingface.co/facebook/nllb-200-3.3B, 2022.
[15] N. Team, M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun, S. Wang, G. Wenzek, A. Youngblood, B. Akula, L. Barrault, G. M. Gonzalez, P. Hansanti, J. Hoffman, S. Jarrett, K. R. Sadagopan, D. Rowe, S. Spruit, C. Tran, P. Andrews, N. F. Ayan, S. Bhosale, S. Edunov, A. Fan, C. Gao, V. Goswami, F. Guzmán, P. Koehn, A. Mourachko, C. Ropers, S. Saleem, H. Schwenk, J. Wang, No language left behind: Scaling human-centered machine translation, 2022. URL: https://arxiv.org/abs/2207.04672. arXiv:2207.04672.
[16] M. L. (saffsd), langid.py: Stand-alone language identification system, https://github.com/saffsd/langid.py, 2011.
[17] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python, 2020. doi:10.5281/zenodo.1212303.
[18] P. Analyzer, Presidio analyzer documentation, https://microsoft.github.io/presidio/analyzer/, 2025.
[19] seatgeek, Thefuzz: Fuzzy string matching in python, https://github.com/seatgeek/thefuzz, 2021.
[20] MongoDB, Mongodb documentation, https://www.mongodb.com/docs/, 2025.
[21] Discord, Discord documentation, https://support.discord.com/hc/en-us, 2025.
[22] Telegram, Isis watch, https://t.me/ISISwatch, 2025. Official Telegram channel providing daily updates on banned terrorist content.
[23] MongoDB, Pymongo documentation, https://www.mongodb.com/docs/languages/python/pymongo-driver/current/, 2025.
[24] MongoDB, Pymongo: The official mongodb python driver, https://github.com/mongodb/mongo-python-driver, 2009.
[25] S. S. Roy, E. P. Vafa, K. Khanmohamaddi, S. Nilizadeh, DarkGram: A large-scale analysis of cybercriminal activity channels on telegram, in: 34th USENIX Security Symposium (USENIX Security 25), 2025, pp. 4839–4858.
[26] D. R. Arikkat, S. Nicolazzo, A. Nocera, et al., CTI dataset construction from Telegram, arXiv preprint arXiv:2509.20943 (2025).
[27] M. G. (fastfire), deepdarkcti: Collection of cyber threat intelligence sources from the deep and dark web, https://github.com/fastfire/deepdarkCTI, 2021.
[28] ItIsMeCall911, Awesome-Telegram-OSINT: A curated list of awesome telegram osint tools, sites and resources, https://github.com/ItIsMeCall911/Awesome-Telegram-OSINT, 2021.
[29] Breachsense, Cti: Telegram threat actor channels, https://www.breachsense.com/threat-actor-channels/, 2025.
[30] Telegago, Telegago: Programmable search engine for telegram, https://cse.google.com/cse?&cx=006368593537057042503:efxu7xprihg#gsc.tab=0, 2025.
[31] Telemetr, Telemetr.io: Analytics service for businesses on Telegram, https://telemetr.io/, 2025.
[32] TGStat, Tgstat.ru: Catalog of telegram channels and chats, https://tgstat.ru/, 2025.
[33] D. Antonakaki, S. Ioannidis, Israel-hamas war through telegram, reddit and twitter, arXiv preprint arXiv:2502.00060 (2025).
[34] C. E. Project, Terrorists on telegram, https://www.counterextremism.com/terrorists-on-telegram, 2017.
[35] D. Siman-Tov, Muslim brotherhood telegram network connected to boulder terrorist, https://www.inss.org.il/social_media/

muslim-brotherhood-telegram-network-connected-to-boulder-terrorist/, 2025.
[36] A. Guerra, M. Lepre, O. Karakuş, Quantifying extreme opinions on reddit amidst the 2023 Israeli-Palestinian conflict, Natural Language Processing Journal (2025) 100156.
[37] Cyberpress, Reddit cybersecurity: A comprehensive list of 45+ subreddits, https://www.cyberpress.io/learn/lists/reddit-cybersecurity/, 2023.
[38] A. O. Academy, Youtube search tool - youtube investigations, https://www.aware-online.com/en/osint-tools/youtube-search-tool/, 2019.
[39] M. W. (mattwright324), youtube-geofind: Web-tool to search youtube for geographically tagged videos by channel, topic, and location. videos are viewable in a map and exportable to csv, https://github.com/mattwright324/youtube-geofind, 2017.
[40] Forbes, Global 2000: The world's largest public companies ranked, https://www.forbes.com/global2000/list/24/#tab:overall, 2024.
[41] Radware, October 7: Post-threat analysis, https://www.radware.com/security/threat-advisories-and-attack-reports/october-7-post-threat-analysis/, 2025.
[42] H. AG, Top middle east cyber threats, https://www.helpag.com/top-middle-east-cyber-threats-october-29th-2025/, 2025.
[43] Meta, Dangerous individuals and organizations, https://transparency.meta.com/policies/community-standards/dangerous-individuals-organizations/, 2025.
[44] S. Biddle, Revealed: Facebook's secret blacklist of "dangerous individuals and organizations", https://theintercept.com/2021/10/12/facebook-secret-blacklist-dangerous/, 2021.
[45] K. Zettl-Schabath, J. Bund, M. Müller, C. Borrett, J. Hemmelskamp, A. Alibegovic, E. Bajra, A. Jazxhi, E. Kellenter, A. Sachs, C. Shelley, Global dataset of cyber incidents, 2025. URL: https://doi.org/10.5281/zenodo.14965395. doi:10.5281/zenodo.14965395.