iJETRM
International Journal of Engineering Technology Research & Management
(IJETRM)
https://ijetrm.com/

AI-POWERED TELEGRAM OSINT TOOL FOR CYBER THREAT DETECTION AND ANALYSIS – REVIEW

VidhyaSathish
Assistant Professor, School of Computing Science,
Department of Advanced Computing and Analytics, VISTAS, Pallavaram, Chennai
vidhyasathish83@gmail.com
V. Queen Jemila
Assistant Professor, Department of Computer Applications,
V. V. Vanniaperumal College for Women, Virudhunagar.
queenjemila@vvvcollege.org
T. Devasena
Assistant Professor, Department of Cognitive Systems,
SDNB Vaishnav college for women, Chrompet, Chennai
devasenasuresh07@gmail.com

ABSTRACT
It is now essential to monitor and examine groups and channels for dangerous information due to the increase in cyber dangers and fraudulent activity on social media, especially on sites like Telegram. The AI-Powered Telegram OSINT Tool project uses OpenAI, NLP, and Telegram scraping to identify phishing attempts, scams, fraud, and dark web activity. This application offers profound insights into questionable actions by fusing real-time keyword-based surveillance, sentiment analysis powered by AI, contextual risk assessment, and graphical trend visualization. OpenAI integration improves accuracy and interaction, allowing users to query hazards and obtain insightful information. The paper is intended to improve threat intelligence for analysts, law enforcement, and cybersecurity experts.

Keywords:
OpenAI, NLP, OSINT Tool, Cyber Threat Detection

INTRODUCTION

Telegram has rapidly expanded into one of the most prominent messaging networks, with over 700 million active users globally. Telegram, which is well-known for its strong encryption, anonymity, extensive broadcasting capabilities, and lax moderation guidelines, has turned into a double-edged sword that not only makes it easier for people to communicate and form communities, but also makes it easier for cybercriminals to flourish. Telegram's services are increasingly being used by cyber adversaries to plan and advance illegal activities. These include disseminating false information and phishing links, conducting extensive investment frauds, disseminating hacking tools, and organizing dark web-related operations like illicit marketplaces and data breaches. Businesses, people, and national security are all seriously threatened by this misuse. Conventional techniques for detecting threats, such manual moderation or basic keyword-based filtering, are no longer adequate. These methods frequently produce a large number of false positives or undetected threats, struggle with obfuscated language or coded threats, and lack contextual knowledge. A more sophisticated, intelligent method of threat detection is required due to the dynamic and unstructured nature of Telegram content. The AI-Powered Telegram OSINT Tool for Cyber Threat Detection and Analysis was created as a proactive way to address this expanding problem. This program is a desktop Python application with a Tkinter GUI that provides cybersecurity experts with an easy-to-use interface.It uses robust Telegram client libraries, such as Pyrogram or Telethon, to scrape and get live messages from open Telegram chats, groups, and channels.

iJETRM
International Journal of Engineering Technology Research & Management
(IJETRM)
https://ijetrm.com/

SYSTEM ANALYSIS

At the moment, Telegram's cyber threat identification mostly depends on a mix of manual labor, platform-level moderation (which is frequently quite light), and a few simple automated techniques. The dynamic and complex nature of cybercriminal activity on the network frequently makes these current solutions insufficient.

Manual Moderation and User Reporting:

Description: Administrators or moderators may oversee material in Telegram groups and channels, removing postings or banning members who seem suspicious or break community rules. Additionally, users can alert platform managers to questionable content.

Restrictions:

Manual moderation is incredibly ineffective and unable to keep up with the amount of content being produced, especially when there are millions of active users and innumerable channels.

Subjectivity: Choices about moderation may be arbitrary and inconsistent.

Delayed Response: Before threats are manually recognized and dealt with, they may spread quickly.

Restricted Scope: A lot of illegal activity takes place in secret groups or channels that are difficult for moderators or the public to view.

Lack of Technical Expertise: Moderators might not have the cybersecurity know-how to spot complex dangers like malware spreading or phishing URLs.

Basic Automated Tools for Keyword-Based Filtering:

Description: Some simple tools may use keyword filtering to automatically detect messages that contain terms like "hack," "scam," "bitcoin," or "password" that are linked to illegal activity.

Restrictions:

Lack of Context: Keyword matching is not contextually aware, which can result in a significant number of false negatives (missing threats that utilize specific keywords or obscure language) and false positives (flagging valid talks).

Evasion Techniques: To get over basic keyword filters, cybercriminals frequently employ synonyms, alternate spellings, or coded language.

Incapacity to Identify Complex Threats: Keyword filtering is unable to detect sophisticated threats such as coordinated disinformation operations or sophisticated phishing attempts.

Limited Public Information (Third-Party Monitoring Services):

Description: A few outside cybersecurity companies might provide social media platform monitoring, possibly including Telegram. However, precise information regarding their efficacy and methodology is frequently confidential and not publicly accessible.

Restrictions:

Cost: These services can be pricey, and not all businesses or people may be able to afford them.

Data Privacy Issues: There may be privacy issues when sharing Telegram data with outside services.

Access Restrictions: Private groups and channels may still be inaccessible.

Platform-Level Automated Moderation (Minimal):

Description: Although Telegram prioritizes privacy and little censorship, it probably has some automated internal methods to identify and eliminate some kinds of unlawful content (such as content that promotes terrorism or child sexual abuse). Nevertheless, the breadth and complexity of these systems for more comprehensive cyber threat detection are not made public and seem to be constrained in comparison to other significant platforms.

Restrictions:

Privacy Focus: The degree to which Telegram actively monitors material may be constrained by its significant emphasis on user privacy.

Emphasis on Extreme Content: It's probable that automated moderation gives more weight to content that is blatantly unlawful than to more subtle online dangers like fraud or misinformation.

PROPOSED SOLUTION: AI-POWERED TELEGRAM OSINT TOOL

The AI-Powered Telegram OSINT Tool aims to overcome the limitations of existing solutions by offering a proactive, intelligent, and user-centric approach to cyber threat detection and analysis on Telegram.

Key Advantages:

● Automated Data Collection: Compared to human monitoring, the program greatly increases productivity by automating the process of extracting messages from public Telegram channels.

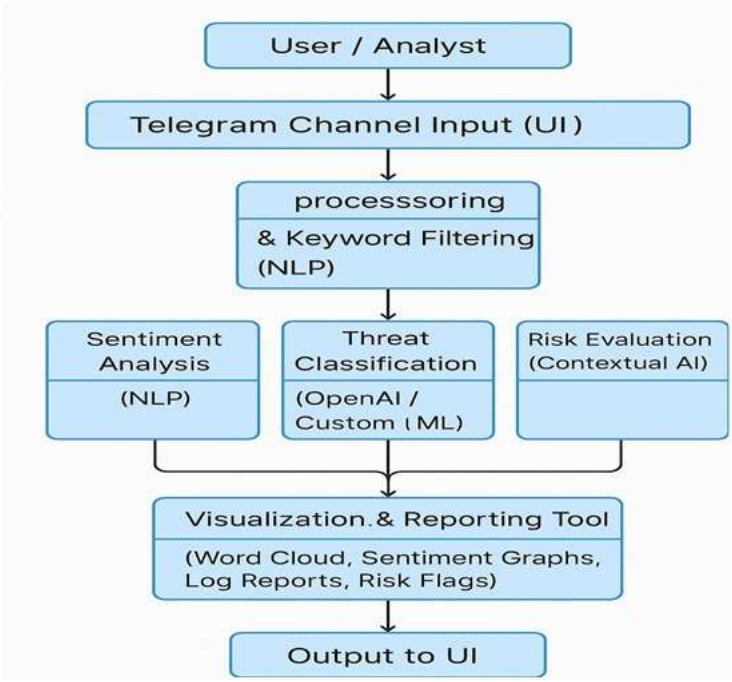
iJETRM
International Journal of Engineering Technology Research & Management
(IJETRM)
https://ijetrm.com/

●Intelligent Analysis (Future Potential): The architecture is made to incorporate more sophisticated Natural Language Processing (NLP) and Machine Learning (ML) techniques in the future, even though the current implementation only employs simple keyword matching. This will make it possible for:
Contextual Understanding: Recognizing dangers not just by keywords but also by the context and meaning of communications. Finding odd patterns or behaviors that could point to malevolent activity is known as anomaly detection. Sentiment analysis is the process of identifying emotionally charged words that are frequently utilized in deception or efforts at manipulation.
Link analysis is the process of locating potentially harmful websites.
● User-Friendly Interface: Professionals in cybersecurity with different degrees of technical proficiency can utilize the application thanks to its graphical user interface.
●Local Operation and Data Privacy: The tool guarantees data privacy and maybe faster processing by running locally rather than depending on external cloud AI services.
●Actionable Insights through Visualization: Pie charts and other visual analytics offer a rapid and simple method of comprehending the frequency of possible hazards in a scanned channel. The "Scam History" provides a log for monitoring and examination.
●Extensibility: New features and more advanced threat detection models can be added in the future thanks to the Python-based design.
●OSINT Focus: The tool is made especially for gathering Open-Source Intelligence, enabling investigators to proactively find and examine publicly accessible data about cyberthreats on Telegram.

How it Addresses Limitations of Existing Solutions:

Scalability: Compared to manual efforts, automation enables the monitoring of a greater number of channels.
●Context and Evasion: By comprehending context and possibly detecting disguised threats, future AI/NLP integration will overcome the drawbacks of keyword-based filtering.
Cost and Accessibility: Because it is a locally installed application, it may be less costly than pricey third-party services.
●Data Privacy: By operating locally, data privacy issues related to sharing data with outside services are lessened.
●Actionable Intelligence: To support analysis and decision-making, the tool offers organized data and visualizations.

Data Flow Diagram (DFD)



iJETRM
International Journal of Engineering Technology Research & Management
(IJETRM)
https://ijetrm.com/

1. **USER LOGIN PROCESS:**

●Input: The user launches the program. The Telegram Login user interface is displayed. The user's phone number is entered.

●Flow: The GUI records the phone number that is input. When you click "Send Code," the Telethon library receives the phone number and uses it to send a code request to the Telegram servers. A verification code is sent to the user's Telegram account via Telegram servers. After receiving the code, the user inputs it into the GUI's "Enter Code" field.

Telethon receives this code and the phone number in order to authenticate against the Telegram servers. The user is asked for their password, which is subsequently sent to Telethon for validation if two-factor authentication is set.

Result:

○Success: Telethon creates a session with the Telegram API when authentication is successful. The Telegram Dashboard appears once the Login user interface has been closed. Through the API, user data (name, username) is obtained from Telegram and shown on the dashboard.

○Failure: The login process resets and the user is presented with an error message in the Login UI if authentication fails (incorrect code, password, or other Telegram API issues).

2. **SCANNING A TELEGRAM CHANNEL FOR SCAMS:**

Input: The user presses the "Scan Channel for Scam" button after interacting with the dashboard. The user is prompted to input the URL or username of the Telegram channel in a dialog box.

Flow: The GUI records the channel identifier that has been entered. To resolve the channel entity (get the internal ID of the channel from Telegram), this identifier is sent to the Telethon library.

The recent message history of the designated channel is then retrieved by Telethon via a request to the Telegram API (up to a configured limit, presently 50). The text content of the mails that were retrieved is extracted. The detect_scams function then receives these notifications. Iteratively going through each message, the detect_scams method looks for predefined scam keywords. These keywords are used to identify messages that may be frauds.Recorded are the total number of messages scanned, the number of scam messages found, and the scam messages' content. This summaries data (safe count, scam count) are used by the display_pie_chart function. The save_scam_history function receives the list of scam messages found and the scanned channel number.

The result:

If any scam messages are detected, a pop-up warning displaying the number of frauds discovered is displayed. If no scams are found, a pop-up informative message appears. The dashboard shows a pie chart that illustrates the percentage of safe and scam communications. The scanned channel and the identified scam messages are recorded in a new entry that is appended to the local scam_history.txt file.

3. **VIEWING SCAM HISTORY:**

- Input: The user clicks the "View Scam History" button on the dashboard.
- Flow:
  - The application checks if the scam_history.txt file exists. If the file exists, its content is read.

Output:

  - A message box is displayed showing the content of the scam_history.txt file. If the file is empty or doesn't exist, a message indicating "No history yet" or "No history found" is shown.

4.

5. **DISPLAYING THE NEWS FEED:**

- Input: Upon successful login and dashboarddisplay, the fetch_news_messages function is automatically called.
- Flow:
  - The function uses Telethon to resolve the entity of a predefined news channel (currently hardcoded).
  - Telethon requests the recent message history (up to 5 messages) from this news channel

iJETRM
International Journal of Engineering Technology Research & Management
(IJETRM)
https://ijetrm.com/

via the Telegram API. The text content of these messages is extracted.

Output:

- The extracted news messages are displayed in the scrolled text area at the bottom of the dashboard. If fetching fails due to network issues or an invalid channel, an error message is displayed in the news feed area.

6. **LOGOUT PROCESS:**

- Input: The user clicks the "Logout" button on the dashboard.
- Flow:
  - The logout function calls the Telethon library's log_out() method to terminate the current Telegram session. The dashboard window is closed.

Output:

- An information message box is displayed confirming that the user has been logged out.

In summary, the data flow involves user input through the GUI, communication with the Telegram API via the Telethon library to retrieve channel and user data, local processing of message content for basic scam detection, visualization of results using Matplotlib, and persistent storage of scan history in a local text file. The news feed functionality involves a similar data retrieval process from a designated channel.

CONCLUSION

A more thorough and automated testing approach will be crucial for guaranteeing the quality, dependability, and security of the AI-Powered Telegram OSINT Tool as it develops and adds more advanced features. Future testing should take into account the following in addition to the initial considerations:

Automated UI Testing: Using automated UI testing frameworks (like PyAutoGUI and Selenium) to mimic user activities and confirm how the graphical user interface behaves in various operating systems and settings. As the user interface is updated, this can help detect regressions and guarantee consistent functionality.

API Testing: creating particular tests to communicate directly with the Telegram API and the Telethon library, confirming that data requests and answers are correct, resolving errors related to API-specific problems (such as rate limits or temporary unavailability), and making sure that sessions are managed correctly.

In-depth Security Testing: Performing comprehensive security evaluations, including static and dynamic code analysis, to find possible weaknesses like data injection risk, unsafe storage of API keys (even locally), and other security flaws. Security professionals may do penetration testing as part of this.

Usability Testing with Target Users: Conducting usability testing sessions with cybersecurity analysts and investigators to acquire their opinions on the tool's overall usability, effectiveness, and intuitiveness. This can assist in pinpointing places where the interface design and process need to be improved.

Cross-Platform Testing: Verifying that the program runs accurately and reliably on all supported desktop environments and operating systems, including Windows, macOS, and Linux.

Regression Testing: Using automated regression test suites to make sure that bug fixes or new features don't unintentionally cause problems or interfere with functionality. As the codebase changes, these tests ought to be performed on a frequent basis.

Data Integrity Testing: Creating tests to confirm that the information included in the scam_history.txt file (as well as any upcoming database implementations) is consistent and intact across various operations and application states.

REFERENCES

1. Bimal Ghimire and Danda B. Rawat. 2022. Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things. IEEE Internet of Things Journal 9, 11 (2022), 8229–8249. DOI: http://dx.doi.org/10.1109/JIOT.2022.3150363
2. Riccardo Ghioni, Mariarosaria Taddeo, and Luciano Floridi. 2023. Open Source Intelligence and AI: A Systematic Review of the Gelsi Literature? AI and Society (2023), 1–16. DOI:http://dx.doi.org/10.1007/s00146-023-01628-x
3. Helen Gibson, Steve Ramwell, and Tony Day. 2016. Analysis, Interpretation and Validation of Open Source Data. Springer International Publishing, Cham, 95–110. DOI:http://dx.doi.org/10.1007/978-3-319-47671-1_7.

iJETRM
International Journal of Engineering Technology Research & Management
(IJETRM)
https://ijetrm.com/

4. Seonghyeon Gong and Changhoon Lee. 2021. Cyber Threat Intelligence Framework for Incident Response in an Energy Cloud Platform. Electronics 10, 3 (2021). DOI:http://dx.doi.org/10.3390/electronics10030239.
5. Gustavo González-Granadillo, Mario Faiella, Ibéria Medeiros, Rui Azevedo, and Susana González-Zarzosa. 2021. ETIP: An Enriched Threat Intelligence Platform for improving OSINT correlation, analysis, visualization and sharing capabilities. Journal of Information Security and Applications 58 (2021), 102715. DOI:http: //dx.doi.org/10.1016/j.jisa.2020.102715
6. Vasile Gheorghit,ă Găitan and Ionel Zagan. 2021. Experimental Implementation and Performance Evaluation of an IoT Access Gateway for the Modbus Extension. Sensors 21, 1 (2021). DOI:http://dx.doi.org/10.3390/s21010246.
7. Hamed Haddadpajouh, Raouf Khayami, Ali Dehghantanha, Kim-Kwang Raymond Choo, and Reza Meimandi Parizi. 2020. AI4SAFE-IoT: an AI-powered secure architecture for edge layer of Internet of things. Neural Computing and Applications (2020), 16119–16133. DOI:http://dx.doi.org/10.1007/s00521-020-04772-3.
8. Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu. 2020. Efficient and Privacy-Enhanced Federated Learning for Industrial Artificial Intelligence. IEEE Transactions on Industrial Informatics 16, 10 (2020), 6532–6542. DOI:http://dx.doi.org/10.1109/TII.2019.2945367