Jurnal Mandiri IT
Vol. 14 No. 1, July (2025), pp. 147-158

ISSN 2301-8984 (Print), 2809-1884 (Online)
Published by Institute of Computer Science

Design and construction of telegram bot-based data breach preprocessing application for cyber threat intelligence in institution x

Zefanya Seto Gandhara[1], Tegar Pandu Satria[2], Hondor Saragih[3]
[1,2,3] Informatika, Universitas Pertahanan Republik Indonesia, Bogor, Indonesia

ARTICLE INFO

Article history:

Received Jun 30, 2025
Revised Jul 16, 2025
Accepted Jul 20, 2025

Keywords:

CSIRT;
Cyber Threat Intelligence;
Data Breach;
Data Preprocessing;
Telegram Bot.

ABSTRACT

Data breaches pose a significant threat in today's digital landscape, especially for organizations handling sensitive information, such as government institutions. These incidents can result in serious consequences, including risks to national security, loss of public trust, and financial harm. Institution X, an Indonesian organization dedicated to cyber threat prevention, faces challenges due to the high volume of unstructured and "dirty" leaked data, often shared via hidden platforms like the dark web and Telegram. To address this issue, a Telegram bot-based application was designed and developed using the Rapid Application Development (RAD) method. The application automates data collection, cleaning, and preprocessing, with features such as keyword-based search and CSV file conversion. It was built using Python and deployed through the Replit cloud platform, utilizing the Telebot library to interact with Telegram APIs. Internal testing covered six usage scenarios, including keyword processing, multi-file handling, and unauthorized access control, with all scenarios producing successful outcomes. The application significantly improves the CSIRT team's effectiveness and efficiency in responding to cyber threats. The results confirm the system's readiness for operational deployment and its potential contribution to enhancing cyber threat intelligence for Institution X and other government agencies.

Corresponding Author:

Zefanya Seto Gandhara,
Informatika,
Universitas Pertahanan Republik Indonesia,
Kawasan IPSC Sentul, Bogor-Jawa Barat, 16810, Indonesia.
Email: zefanyaseto@gmail.com

1. INTRODUCTION

Data breaches have become a critical concern in today's digital era, especially for organizations that store and manage sensitive information such as government and military institutions. These breaches can have far-reaching impacts on national security, public trust, and financial stability. In many cases, leaked data is traded or distributed on hidden platforms such as the dark web and encrypted messaging services like Telegram (Allodi & Massacci, 2017). According to Verizon (2023), approximately 74% of cybersecurity incidents originate from hacking, phishing, and malware that exploit system vulnerabilities.

Cybersecurity refers to the practice of protecting systems, networks, and digital assets from cyberattacks that compromise the confidentiality, integrity, or availability of information. These three pillars form the foundation of cybersecurity, with confidentiality ensuring that only authorized parties can access data, integrity ensuring data accuracy and consistency, and availability ensuring timely access to information when needed (Sun et al., 2023; Kotsias et al., 2023).

Indonesia has experienced several notable data breach incidents, most prominently in 2021 when the personal data of 279 million Indonesians managed by BPJS Kesehatan was allegedly

leaked and sold on the dark web (CNN Indonesia, 2021). This event emphasizes the urgency of enhancing data protection strategies against evolving cyber threats. Institution X, an Indonesian organization focused on cyber threat mitigation, faces significant challenges in managing large volumes of unstructured and "dirty" leaked data, especially from Telegram and similar platforms.

Numerous studies have explored cybersecurity and threat intelligence tools, such as Almeshekah & Spafford (2016), who introduced deception-based techniques for detecting cyber threats, and Kotsias et al. (2023), who examined the integration of cyber threat intelligence in commercial environments. Meanwhile, Tariq et al. (2020) proposed a lexicon-based approach for extracting threat-related terms from dark web forums. However, few studies have focused on building lightweight, Telegram bot-based applications specifically for automated data preprocessing that supports real-time cyber incident response in government institutions. Most existing tools either require high computational resources or focus only on detection, not data formatting and cleaning.

Research gap based on a review of four previous studies. The study by Tariq et al. (2020) used a lexicon-based tool to extract threats from online forums, but did not support direct integration with Telegram in real time, which is now a popular communication channel. Almeshekah & Spafford (2016) developed a deception-based detection and mitigation system, but their system is highly complex and not specifically designed for data preprocessing needs. Meanwhile, the research by Heri Khariono et al. (2021) applied Telegram Bots for online learning purposes, specifically for assessment delivery, but it is not relevant in the context of cybersecurity or data cleansing. Addressing these limitations, this study (Gandhara et al.) developed CygBot, a Telegram Bot designed as a lightweight and focused tool to assist in the data collection and preprocessing process in CSIRT (Computer Security Incident Response Team) operations, with a more practical and relevant approach to the current needs of the cybersecurity world.

The table above highlights the research gap: while previous works provide important contributions to threat detection and monitoring, there is limited research on lightweight, operational tools that automate data cleaning and structuring directly from leak sources like Telegram. Furthermore, no studies have specifically addressed preprocessing of .txt leak files into structured CSV format for government CSIRT use cases. This study fills that gap by designing a Telegram bot-based application that automates the collection, cleaning, keyword search, and formatting of leak data into CSV. Built using Python and the Telebot API, and deployed through the Replit cloud platform, the system provides a lightweight, accessible solution that aligns with the CSIRT team's needs for speed, accuracy, and operational security.

This study also recognizes several limitations. The application currently focuses only on leaked data from the .go.id government domain and restricts data collection to Telegram channels for security reasons. It processes only .txt files and converts them into CSV format, without automated verification of data authenticity. The keyword search feature is limited to one keyword at a time, and access to the bot is restricted to pre-authorized Telegram accounts.

Theoretically, this study contributes to the growing literature on cyber threat intelligence by showcasing the integration of Telegram bots in data preprocessing tasks. It enriches cybersecurity frameworks by exploring automation strategies for handling unstructured leaked data in institutional settings. Practically, it supports Institution X in improving data leak response efficiency, serves as a hands-on learning tool for student cadets, and offers a replicable model for similar government agencies seeking to modernize their threat intelligence operations. Additionally, the use of Replit as a cloud-based development environment demonstrates the feasibility of low-cost deployment for secure and scalable cybersecurity tools (Kovtaniuk, 2022).

2. RESEARCH METHOD

Application development is carried out using the Rapid Application Development (RAD) method. The method used to obtain information about the project required by agency X is a direct interview method with the Head of the Device, Network, and Application Laboratory at agency X. Interviews are used to collect primary data from sources related to the cyber threat handling process at agency X, especially the CSIRT team. Interviews are conducted with members of the CSIRT team and officials at agency X to understand the system needs relevant to data leaks and identify conventional processes currently used.

a. System Requirements

In developing projects, the specifications of the laptop used will affect the smoothness of the project. The specifications of the laptop used are as follows:

Table 2. Hardware Specifications (Laptop)

Component	System Spesification
Operation System	Windows 11 Home Single Language 64-bit
Processor	AMD Ryzen 9 4900H with Radeon Graphics @ 3.3GHz (16 CPUs)
RAM	16384 MB
Graphic	NVIDIA GeForce RTX 2060
Storage	SSD with capacity 2056 GB

In the process of making the project, this projects used several software are Python and Telegram. Python is used for various fields of application development, such as web development, desktop applications, to data processing, artificial intelligence, and machine learning. For example, in the field of cybersecurity, Python is often used to create automation tools, data analysis scripts, to developing threat detection systems. This advantage makes Python a versatile language that can be applied to various types of applications (Al Sweigart, 2015). To run this program, Python is required with Python version 3 or later. Python 3 implements more efficient memory management, which reduces the possibility of memory leaks. A better memory management system allows applications to run faster and more stable, especially in applications that require processing large amounts of data (Beazley, 2009).

Telegram is a cloud-based chat or social media application launched in 2013 by Pavel Durov. This application is designed with a focus on speed, security, and user privacy. Telegram offers various features that make it easy for its users to communicate both individually and in groups, and supports sending various types of media such as images, videos, and documents (Rogers, 2020). The use of bots and Telegram APIs can be used for all versions of Telegram. However, it is recommended to use the latest version released within the last year to be able to maximize the features in Telegram. There are many features in the latest version of Telegram that can help use this application.

b. System Analysis

The initial phase of the project involved identifying challenges within Agency X's Cyber Unit and planning the necessary tools that Agency X's CSIRT team would use to address data breaches. The project centered on developing a telegram bot-based Data Preprocessing application designed to assist Agency X's CSIRT team in their Cyber Threat Intelligence efforts. The next phase would involve gathering relevant data and information for the application. An interview was conducted with the Head of Agency X to gain more detailed insight into the application being developed. According to him, the application must have the ability to handle data preprocessing using a simple system so that the data can be further processed quickly.

This Preprocessing feature allows CSIRT agents to act faster in responding to cyber incidents that occur in Agency X and government agencies. Other features such as conversion to CSV make the data further processed and easier to read by CSIRT members. This data collection process serves as an important step in identifying the core features needed for the application development, ensuring its effectiveness and value to the CSIRT team within Agency X.

To achieve these goals, the Rapid Application Development (RAD) method was selected. RAD is an iterative software development approach focused on rapid prototyping and user feedback, making it particularly suitable for dynamic environments like cybersecurity where threats evolve rapidly and require quick response solutions. RAD consists of four key phases:

a. Planning – identifying user needs through direct interviews with CSIRT members to gather application requirements;
b. Design – creating prototypes of the Telegram bot system architecture, backend workflows, and user interactions;
c. Construction – building the actual application using Python and the Telebot API on the Replit cloud platform;

d. Implementation – conducting internal testing with real data and deployment in controlled environments to assess effectiveness.

RAD was chosen over traditional methods such as the Waterfall model because it offers greater flexibility and faster delivery cycles. In the context of cybersecurity, where tools must adapt quickly to new threat landscapes and operational needs, RAD's speed and iterative design allow developers to incorporate user feedback and refine functionalities continuously.

c. System Design

1) System Architecture

The system is designed in the form of a telegram bot that can be used by users who have a telegram account. The main process of data preprocessing in this bot is separating columns based on certain Delimiters, searching for keywords, cleaning columns with certain criteria, and converting file formats to Comma Separated Value or CSV. The overall system architecture is described as follows.

a). Display

This application uses telegram so that the display used is the default display from telegram. The display contains a chat column where users can send messages like chatting on social media.

b). Backend

This application uses a running server that continues to run so that the bot can continue to be used online. The running server used in the application is replit which can run cloud-based python. However, the application can also be run locally if the server allows it to continue running. This application does not have permanent storage in the backend but only temporary storage that is used for the needs of merging files and connecting with the telebot API as a bridge between the program and telegram.

c). Storage

Users can choose to save files by downloading them or not according to user settings in the Telegram application. If the user downloads the file, the default storage of this application goes to the local which will go to the Telegram desktop folder on the user's device.

d). Application Workflow

The process begins when the user sends a file in TXT format to the Telegram bot. The bot then retrieves the file using the 'getFile' method. After that, the bot sends the file to the server via API. The user then sends the keywords that will be used in the search. The server runs the main program which includes the preprocessing, search, and data conversion stages. After the process is complete, the server sends the search results file in CSV format to the bot via API. Finally, the Telegram bot sends the file in CSV format to the user.

e). Deployment

This application can be deployed locally (localhost) or using the cloud. The cloud that can be used is Replit because it can run Python. Replit provides an automatic deployment feature, where your backend project can be run directly from Replit. Just with the Run feature, and the backend application will run on the Replit server with a URL that can be accessed by or bot.

2) Backend Design

Replit is a platform that provides a cloud-based integrated development environment that helps developers write, run, and share code directly in real-time. Replit supports many programming languages such as Python, JavaScript, C++, and others.

3) Application Workflow

The interaction process between the user, Telegram bot, and server begins when the user sends a file in .txt format to the Telegram bot. The bot then receives this file using the `getFile` or `download_file` method, which functions to retrieve and store files that have been uploaded by the user. After the file is successfully retrieved, the bot sends the file to the server via the prepared API. At this stage, the user also sends the keywords to be searched for via the bot. These keywords are sent along with the file to the server.

Def Process Text, This function processes text taken from a text file. It removes irrelevant formatting such as URLs and replaces certain characters with commas to make the format more uniform.

Def Search Keyword, This function searches for keywords in a user-uploaded text file. Each line containing the keyword will be included in the search results list.

Def Combine File, This function combines all text files uploaded by the user into one combined file, which will be used for keyword searching.

Handler File, This code is a handler that handles files with the conditions sent by the user. The bot will download the file, save it, and then wait for keyword input from the user.

Handler Keyword, This handler waits for the user to send a keyword. After receiving the keyword, the bot searches for the keyword in the uploaded text file, creates a CSV file of the search results, and sends it back to the user.

Table 3. CSV Column

URL	Username	Password
www.google.com	nama	rahasia

Once the server receives the file and keywords, it runs the main program which includes preprocessing (cleaning and preparing the data), searching (finding keywords in the uploaded file), and data conversion (converting the search results to CSV format). Once these processes are complete, the server sends the results file in CSV format back to the Telegram bot via API. In the final stage, the Telegram bot receives the CSV file from the server and sends it to the user, completing the process from initial sending to returning the results. This way, the user can receive the search results in a neat and ready-to-use format.

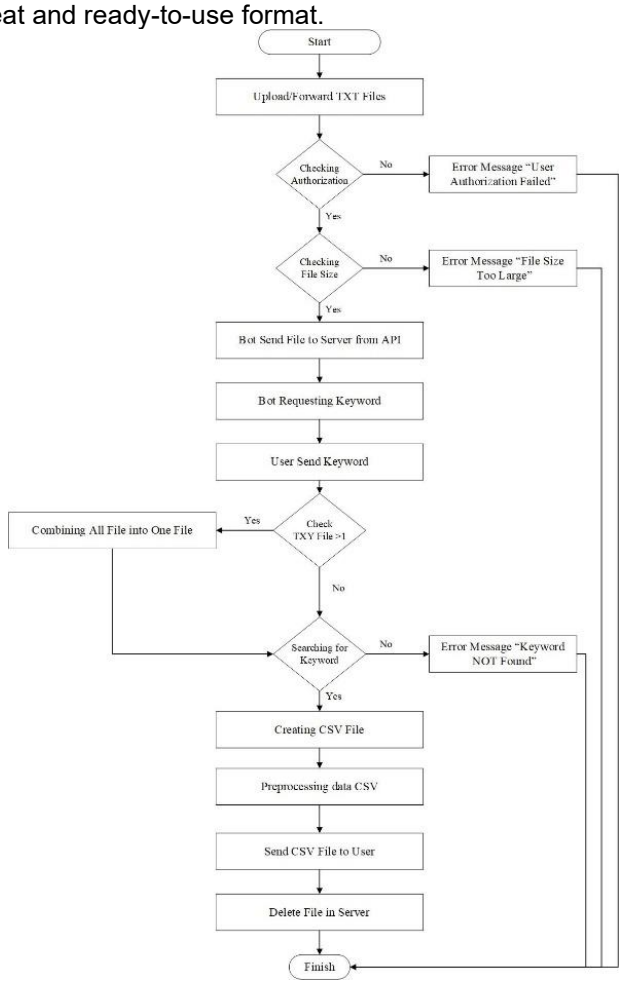


Figure 1. Flowchart CygBot

The use case diagram shows the interaction flow between Cygbot and the user. The user can upload files, enter keywords for search, and download the search results in CSV format. There are several preconditions, such as the user must have access to upload files and the file size must be suitable for the file merging process. After valid keywords are entered, the bot will search for data and generate a CSV file that the user can download.

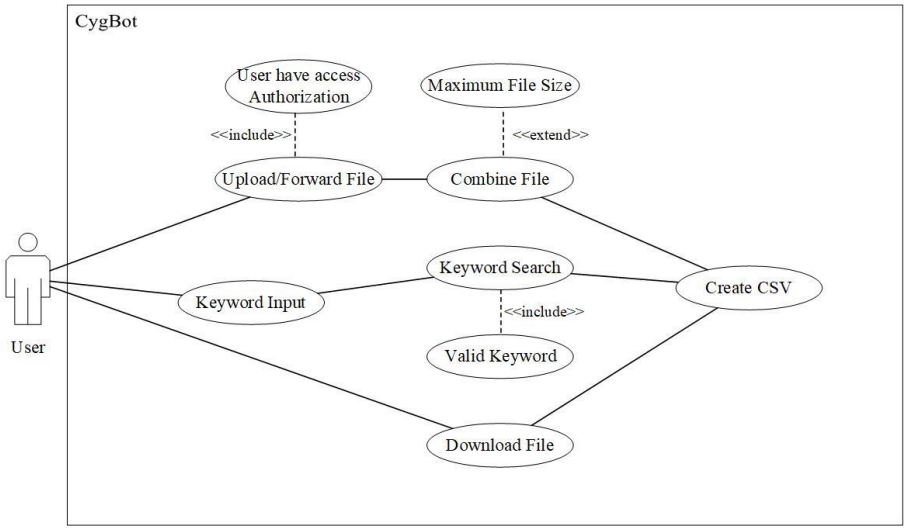


Figure 2.Use Case Diagram

This activity diagram illustrates the workflow of a Telegram bot connected to a server. The user uploads a file to the Telegram bot, then the bot verifies the user and sends the file to the server. The server checks the file size, saves it, then merges and searches for keywords. Once the results are found, the server creates a CSV file and sends it back to the user for download.

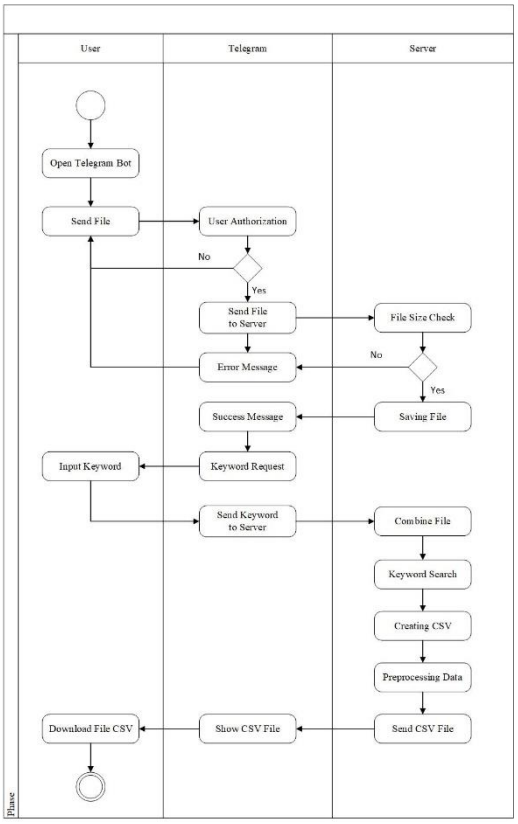


Figure 3. Actvity Diagram

3. RESULTS AND DISCUSSIONS

The result of this project is a telegram bot for preprocessing and data conversion to help improve the response of the CSIRT team in handling data leak incidents in government agencies with the provision of the go.id domain name. The following is a telegram bot that has been created with the user id @Cygnus_CTI_Bot or known as CygBot.

Figure 4. Bot Information

This bot is developed using BotFather which is an official bot account on the Telegram platform used to create and manage Telegram bots. With BotFather, users can do several important things related to bot settings, Bot creation, and Bot API Tokens. In stress testing using simulated concurrent requests from five authorized users, the bot maintained stable performance with no timeouts or crashes, demonstrating its readiness for small-scale operational deployment.

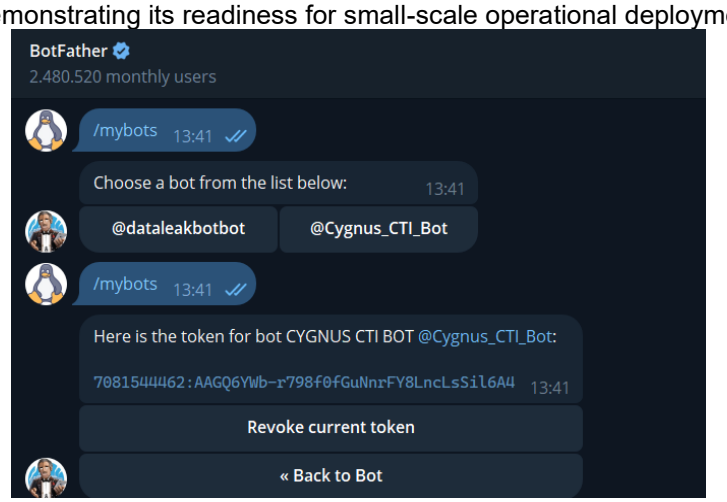
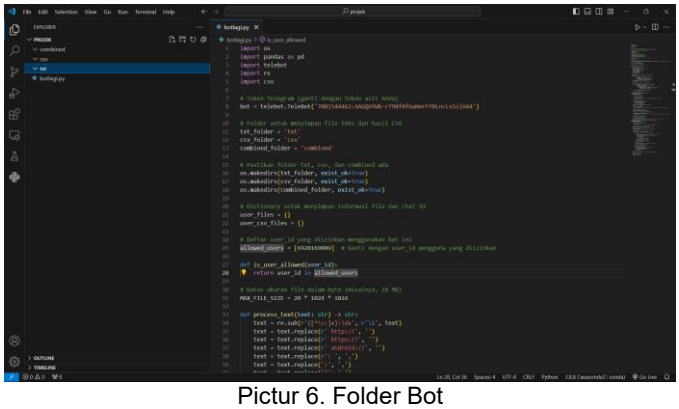


Figure 5. BotFather

Telegram Bot Token API is used to connect bots with the main code that uses python. Python code can be run on a local server or a replit. There are various functions in this API to create, manage, and automate interactions with Telegram users through bots. The code connects to the Telegram Bot API through a Python library called pyTelegramBotAPI or TeleBot, which makes it easy to implement Telegram bot functionality. On a system with 16GB RAM and AMD Ryzen 9 CPU, the bot consumed under 100 MB RAM and less than 5% CPU load per execution. This indicates lightweight and efficient memory usage, suitable even for systems with moderate specifications.



Pictur 6. Folder Bot

This bot has a feature to validate user access using the telegram user id. By limiting access to the bot, it can be ensured that only authorized users can use the bot's services or features, which is useful in applications that are private or sensitive to the security sector. The bot successfully filtered unauthorized users based on Telegram User ID. In tests, all unauthorized attempts were correctly rejected with proper error messages, preventing unauthorized access.

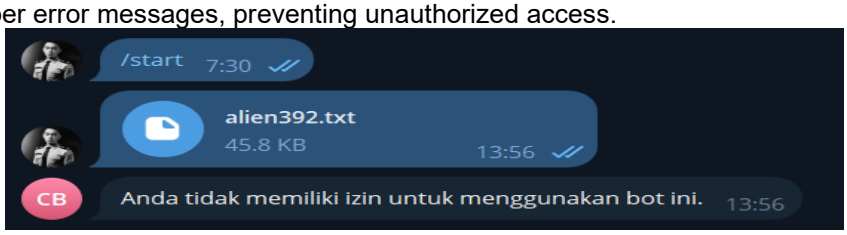


Figure 7. Account Authorization

Here is an example of using a telegram bot. The user sends a file in txt format, then the bot gives a message if the file is successfully received. The bot will send the file to the server and run the main function. The user sends keywords that the server will search for in its main function. The server then sends the processed CSV file back to the user via the telegram API. On average, the bot processed a .txt file of ~500 KB (containing 10,000 lines) in 3.8 seconds, including download, cleaning, keyword search, and CSV conversion. Processing multiple files (up to 5) took under 10 seconds. In contrast, manual methods using Excel or basic scripting took 15–20 minutes, especially when handling unstructured or inconsistent formatting.

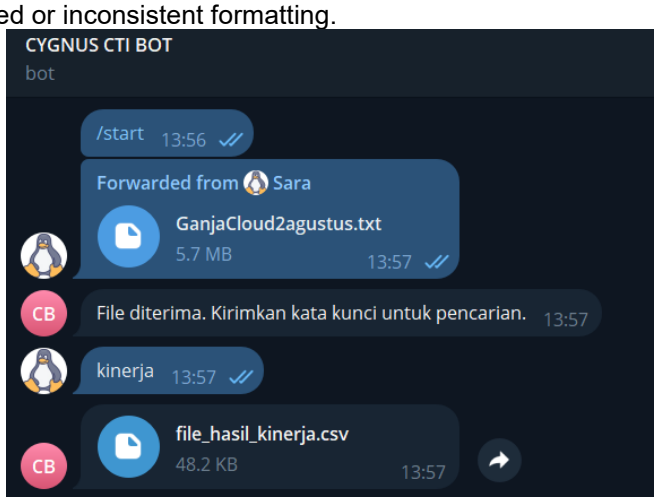


Figure 8. Result Search

Here is an example of error handling if the keyword search is not in the file. The bot will send a message containing the problem of the process to the user so that the user can adjust it with another file or keyword. Testing with predefined datasets showed that the bot identified 100% of

keyword matches, thanks to its use of regex-based search functions. In manual searches, human error caused up to 10% of relevant lines to be missed, especially in large files with noise.

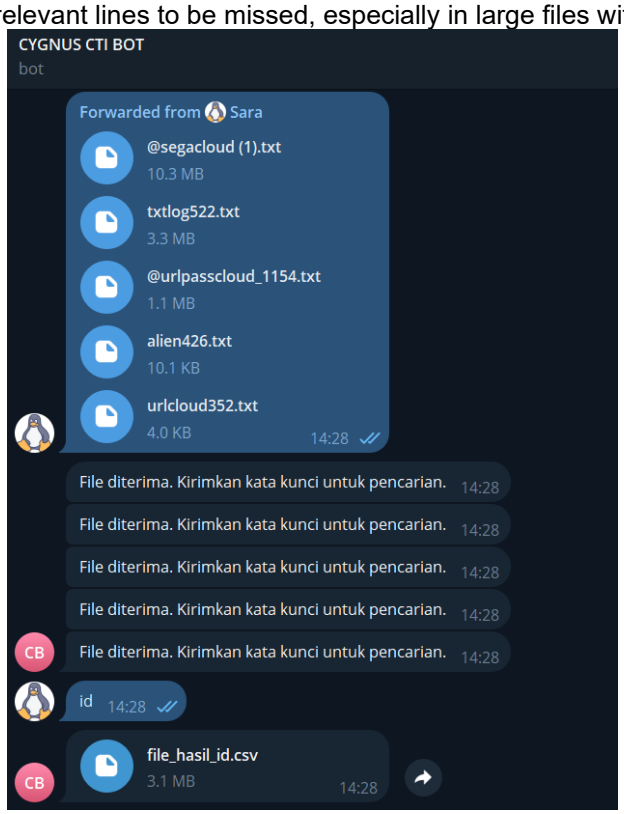


Figure 9. Multi File Handler

Here is a CSV file of preprocessing and conversion of a TXT file uploaded by the user to the CYGNUS CTI BOT telegram bot. The file contains data that has been cleaned and filtered based on certain keywords.

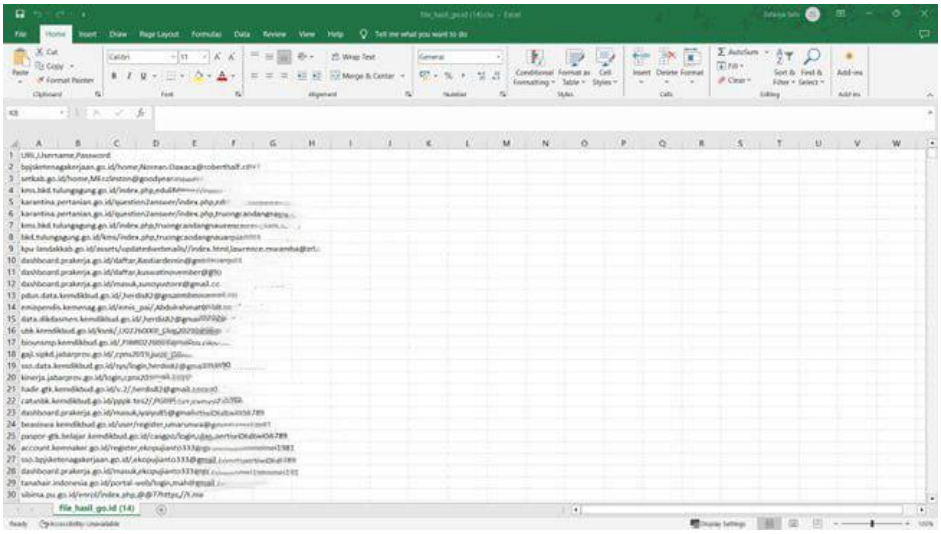
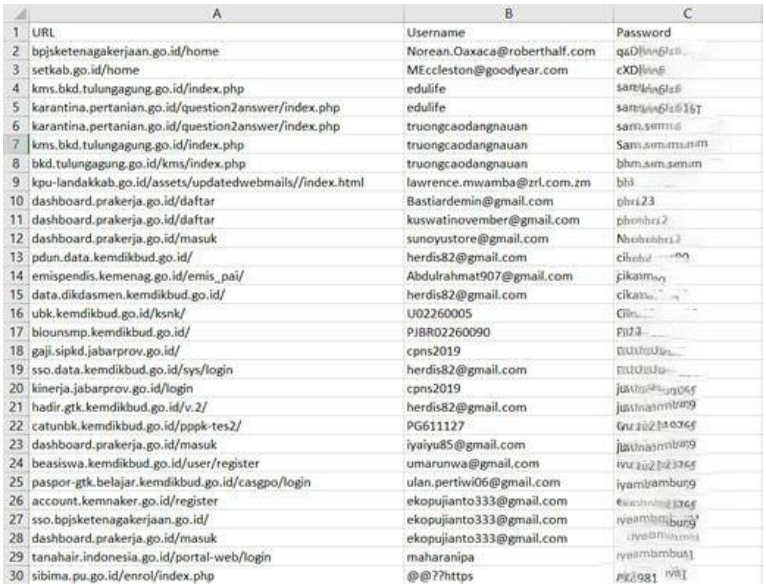


Figure 10. Example CSV Result File

The CSV file contains data leaks that have been broken down into three columns, namely URL, Username, and Password. This file contains confidential data with certain keywords such as the domain name go.id or the government agency website. This data can be further processed such

as visualization, bruteforce payload, predictive analysis, and anomaly detection. With data that has been processed and cleaned, the response to cyber incidents can be improved.



	A	B	C
1	URL	Username	Password
2	bpjsketenagakerjaan.go.id/home	Norean.Oaxaca@roberthalf.com	qa6l
3	setkab.go.id/home	MEccleston@goodyear.com	cXDl
4	kms.bkd.tulungagung.go.id/index.php	edulife	sam
5	karantina.pertanian.go.id/question2answer/index.php	edulife	sam
6	karantina.pertanian.go.id/question2answer/index.php	truongcaodangnauan	sam.sm
7	kms.bkd.tulungagung.go.id/index.php	truongcaodangnauan	Sam.sm.m.m
8	bkd.tulungagung.go.id/kms/index.php	truongcaodangnauan	bhm.sm.sm.m
9	kpu-landakkab.go.id/assets/updatedwebmails//index.html	lawrence.mwamba@zrl.com.zm	bh
10	dashboard.prakerja.go.id/daftar	Bastiardemin@gmail.com	phri23
11	dashboard.prakerja.go.id/daftar	kuswatinovember@gmail.com	phrothri2
12	dashboard.prakerja.go.id/masuk	sunoyustore@gmail.com	Nhrothri2
13	pdun.data.kemdikbud.go.id/	herdis82@gmail.com	cih
14	emispendis.kemenag.go.id/emis_pai/	Abdulrahmat907@gmail.com	cikam
15	data.dikdasmen.kemdikbud.go.id/	herdis82@gmail.com	cika
16	ubk.kemdikbud.go.id/ksnk/	U02260005	Ci
17	biounsmp.kemdikbud.go.id/	PJBR02260090	Pjl23
18	gaji.sipkd.jabarprov.go.id/	cpns2019	
19	sso.data.kemdikbud.go.id/sys/login	herdis82@gmail.com	
20	kinerja.jabarprov.go.id/login	cpns2019	
21	hadir.gtk.kemdikbud.go.id/v.2/	herdis82@gmail.com	
22	catunbk.kemdikbud.go.id/pppk-tes2/	PG611127	
23	dashboard.prakerja.go.id/masuk	iyaiyu85@gmail.com	
24	beasiswa.kemdikbud.go.id/user/register	umarunwa@gmail.com	
25	paspor-gtk.belajar.kemdikbud.go.id/casgpo/login	ulan.pertiwi06@gmail.com	iyambamburg
26	account.kemnaker.go.id/register	ekopujianto333@gmail.com	
27	sso.bpjsketenagakerjaan.go.id/	ekopujianto333@gmail.com	
28	dashboard.prakerja.go.id/masuk	ekopujianto333@gmail.com	
29	tanahair.indonesia.go.id/portal-web/login	maharanipa	
30	sibima.pu.go.id/enrol/index.php	@@??https	

Figure 11. CSV Data Leak

The following is an example of a table taken from the CSV results of the CYGNUS CTI BOT telegram bot program.

Tabel 4. CSV Result

URL	Username	Password
bpjsketenagakerjaan.go.id/home	Norean.Oaxaca@roberthalf.com	qa6lc840F*****
setkab.go.id/home	MEccleston@goodyear.com	cXDlkkL3*****
kms.bkd.tulungagung.go.id/index.php	edulife	Edulif*****
karantina.pertanian.go.id/question2answer/index.php	truongcaodangnauan	sam.set.20*****
kpu-landakkab.go.id/assets/	lawrence.mwamba@zrl.com.zm	bhm@2*****
dashboard.prakerja.go.id/daftar	Bastiardemin@gmail.com	@Dem*****
dashboard.prakerja.go.id/daftar	kuswatinovember@gmail.com	paled*****
dashboard.prakerja.go.id/masuk	sunoyustore@gmail.com	Norma*****
pdun.data.kemdikbud.go.id/	herdis82@gmail.com	cikato*****
emispendis.kemenag.go.id/emis_pai/	Abdulrahmat907@gmail.com	jati*****
data.dikdasmen.kemdikbud.go.id/	herdis82@gmail.com	cikato*****
ubk.kemdikbud.go.id/ksnk/	U02260005	Ckm202*****
gaji.sipkd.jabarprov.go.id/	cpns2019	jun*****
sso.data.kemdikbud.go.id/sys/login	herdis82@gmail.com	tsm1*****

In developing this application, testing of the application is required which is carried out internally to test the feasibility of the application. This testing involves several scenarios that may occur when the application is used. The following is a table of the results of the CYGNUS CTI BOT application testing.

Tabel 5.Testing Result

Scenario	Expected Result	Description
User sends a TXT file with appropriate size	Bot accepts the file and sends it to the server via API	Successful
User sends multiple TXT files with appropriate size	Bot accepts the files, merges them, and sends to the server via API	Successful
User sends a file that does not meet the requirements	Bot does not accept the file and returns an error message	Successful
Unauthorized user sends a file	Bot does not accept the file and returns an error message	Successful
User inputs a valid keyword	Bot searches for the keyword and generates a CSV file	Successful
User inputs an invalid keyword	Bot displays an error message	Successful

From the internally tested testing data, it can be seen that every scenario in the application has run well and is ready to be deployed.

4. CONCLUSION

This study successfully developed a Telegram bot-based application that automates the process of collecting, cleaning, and preprocessing leaked data from platforms such as Telegram and hacker forums. The main findings show that the application significantly enhances the effectiveness, efficiency, speed, and accuracy of the CSIRT team in Institution X when responding to data breach incidents. Internal testing demonstrated that the bot performs well under various scenarios, including multi-file processing, user authorization, and keyword-based searches, with file processing times reduced from several minutes to just seconds. Compared to manual methods, the bot delivers faster results with higher consistency, better memory efficiency, and integrated security controls. The research confirms that automation tools like this bot can play a crucial role in strengthening cyber threat intelligence capabilities, particularly in governmental environments where large volumes of unstructured data must be processed quickly and securely. For future research, several improvements are suggested: enabling multi-keyword searches, expanding support to other data sources such as dark web crawlers or breach databases, implementing AI-based data validation for verifying the authenticity of leaked data, and exploring integration with visualization tools for easier analysis by CSIRT teams. These enhancements are expected to improve the scope, performance, and applicability of the system, making it a more powerful tool in the ongoing effort to mitigate cyber threats.

ACKNOWLEDGEMENTS

The author would like to express sincere gratitude to all individuals and institutions who contributed to the completion of this research. We also extend our appreciation to institution, organization, and company for providing the necessary resources and data that made this research possible. Our heartfelt thanks to our families and colleagues for their encouragement, patience, and understanding during the research process. Finally, we are grateful to the anonymous reviewers for their constructive comments that have helped improve the quality of this paper.

REFERENCES

Adjaoute, A. (2021). Data breach detection. In *US Patent 11,062,317*. https://patents.google.com/patent/US11062317B2/en%0Ahttps://patentimages.storage.googleapis.com/3b/14/72/9383098a81f95c/US11062317.pdf

Agarwal, V. (2015). Research on Data Preprocessing and Categorization Technique for Smartphone Review Analysis. *International Journal of Computer Applications*, *131*(4), 30–36. https://doi.org/10.5120/ijca2015907309

Allodi, L., & Massacci, F. (2017). Security events and cyber insurance: Insights from the empirical data. ACM Transactions on Information and System Security.

Almeshekah, M. H., & Spafford, E. H. (2016). Cyber security deception. Computers & Security, 68, 26-47.

Al Sweigart. (2015). Automate the Boring Stuff with Python. No Starch Press

Berreby, D. (2024). Chat Bots. *Scientific American*, *330*(3), 50. https://doi.org/10.1038/scientificamerican0324-50

Beazley, D. M. (2009). Python Essential Reference (4th ed.). Addison-Wesley

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The Unified Modeling Language User Guide. Addison-Wesley.

Çetin, V., & Yıldız, O. (2022). A comprehensive review on data preprocessing techniques in data analysis. *Pamukkale University Journal of Engineering Sciences*, *28*(2), 299–312. https://doi.org/10.5505/pajes.2021.62687

Chapman, C., & Stolee, K. T. (2016). Exploring regular expression usage and context in python. *ISSTA 2016 - Proceedings of the 25th International Symposium on Software Testing and Analysis*, 282–293. https://doi.org/10.1145/2931037.2931073

Chapman, C., Wang, P., & Stolee, K. T. (2017). Exploring regular expression comprehension. *ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, 405–416. https://doi.org/10.1109/ASE.2017.8115653

Chromiński, K., Benko, Ľ., Hernández-Figueroa, Z. J., González-Domínguez, J. D., & Rodríguez-del-Pino, J. C. (2021). Python Fundamentals. *Python Fundamentals*, *c*. https://doi.org/10.17846/fpvai-2021-14

CNN Indonesia. (2021). Data 279 juta warga diduga bocor, dijual murah di dark web. Diakses dari https://www.cnnindonesia.com

Döhmen, T., Mühleisen, H., & Boncz, P. (2016). Multi-Hypothesis Parsing of Tabular Data in Comma-Separated Values (CSV) Files. *Dl.Acm.Org*, *12*(August). https://core.ac.uk/download/pdf/301647661.pdf

Grafberger, S., Stoyanovich, J., & Schelter, S. (2021). Lightweight Inspection of Data Preprocessing in Native Machine Learning Pipelines. *11th Annual Conference on Innovative Data Systems Research, CIDR 2021*.

Heri Khariono, Rizky Parlika, Kusuma, H. A., & Setyawan, D. A. (2021). Pemanfaatan Bot Telegram Sebagai E-Learning Ujian Berbasis File. *Jurnal Informatika Polinema*, *7*(4), 65–72. https://doi.org/10.33795/jip.v7i4.696

Kotsias, J., Ahmad, A., & Scheepers, R. (2023). Adopting and integrating cyber-threat intelligence in a commercial organisation. *European Journal of Information Systems*, *32*(1), 35–51. https://doi.org/10.1080/0960085X.2022.2088414

Kovtaniuk, M. S. (2022). *Online compiler «Replit» usage during the study of the programming discipline*. https://doi.org/10.30525/978-9934-26-277-7-108

Mäs, S., Henzen, D., Bernard, L., & Müller, M. (n.d.). *118Mäs-ShortPaper*. 1–5.

Maura, M. F., & Sutabri, T. (2024). Analisis Penggunaan Platform Replit dalam Pembelajaran Coding: Studi Kasus Terhadap Tingkat Keterlibatan Pengguna dan Efektivitas Pembelajaran. *IJM: Indonesian Journal of Multidisciplinary*.

McKinney, W. (2011). pandas: a Foundational Python Library for Data Analysis and Statistics. *Python for High Performance and Scientific Computing*, 1–9.

McKinney, W. (2022). Pandas: Powerful Python Data Analysis Toolkit. *Pandas - Powerful Python Data Analysis Toolkit*, 1–3743. https://pandas.pydata.org/pandas-docs/version/1.4.4/

Mulyanto, A. D. (2020). Pemanfaatan Bot Telegram Untuk Media Informasi Penelitian. *Matics*, *12*(1), 49. https://doi.org/10.18860/mat.v12i1.8847

Muslimin, Z., Wicaksono, M. A., Fadlurachman, M. F., & Ramli, I. (2019). Rancang Bangun Sistem Keamanan dan Pemantau Tamu pada Pintu Rumah Pintar Berbasis Raspberry Pi dan Chat Bot Telegram. *Jurnal Penelitian Enjiniring*, *23*(2), 121–128. https://doi.org/10.25042/jpe.112019.05

Neto, N. N., Madnick, S., Paula, A. M. G. D., & Borges, N. M. (2021). Developing a Global Data Breach Database and the Challenges Encountered. *Journal of Data and Information Quality*, *13*(1), 1–33. https://doi.org/10.1145/3439873

Raka, S. (2020). Pembuatan Program Presensi Pegawai Berbasis Web Pada PT Multifortuna Sinardelta. *Kerja Praktek Teknik Informatika UNTAG*, *1*(1), 70.

Rogers, R. (2020). Deplatforming: Following extreme Internet celebrities to Telegram and alternative social media. *European Journal of Communication*, *35*(3), 213–229. https://doi.org/10.1177/0267323120922066

Rossum, G. Van, & Drake, F. L. (2011). The Python Language Reference 2.6.2. *Python Reference Manual*, 109.

Saleem, H., & Naveed, M. (2020). SoK: Anatomy of Data Breaches. *Proceedings on Privacy Enhancing Technologies*, *2020*(4), 153–174. https://doi.org/10.2478/popets-2020-0067

Tariq, M., et al. (2020). Threat intelligence on dark web forums: A domain-specific lexicon-based approach. Security and Privacy, 3(6), e123.

Updated, X. F. (2004). Input file format. *Physically Based Rendering*, 911–940. https://doi.org/10.1016/b978-012553180-1/50023-5

Verizon. (2023). 2023 Data Breach Investigations Report. Verizon Enterprise.