A Scalable Telegram-Based Botnet Framework for Stealthy Remote Command and Control

DOI: https://doi.org/10.54654/isj.v2i25.1102

Pham Van Toi*, Nguyen Trung Dung, Hoang Linh Phuong, Nguyen Huu Long

***Abstract*— In this study, we analyze how Telegram Bots can be abused as a Command and Control (C2) infrastructure in cyberattacks. We propose a Telegram-based C2 model that enables attackers to control compromised systems without relying on a centralized server, thereby enhancing anonymity and evasion capabilities. Furthermore, we introduce detection and defense strategies based on network behavior monitoring and encrypted message analysis.**

***Tóm tắt*— Trong nghiên cứu này, chúng tôi phân tích cách Telegram Bot có thể bị lạm dụng như một hạ tầng điều khiển và chỉ huy (Command and Control - C2) trong các cuộc tấn công mạng. Nhóm tác giả đề xuất một mô hình Telegram - C2 giúp kẻ tấn công điều khiển hệ thống bị nhiễm mà không cần máy chủ trung gian, tăng khả năng ẩn danh và khó bị phát hiện. Hơn nữa, nhóm tác giả cũng đề xuất các biện pháp phát hiện và phòng chống dựa trên giám sát hành vi mạng và phân tích tin nhắn mã hóa.**

***Keywords*— social media; command and control; botnet; information security.**

***Từ khóa*— mạng xã hội; điều khiển và chỉ huy; botnet; an toàn thông tin.**

I. INTRODUCTION

In recent years, there has been a significant rise in the exploitation of legitimate platforms for malicious purposes within the cyber threat landscape. Among these platforms, encrypted messaging applications particularly Telegram have become highly attractive for adversaries due to their strong security mechanisms, censorship resistance, and a robust and accessible Application Programming Interface (API) [1]. The use of Telegram as a covert Command and Control (C2) channel enables attackers to remotely operate infected hosts without relying on traditional server-based infrastructures [2], thereby enhancing operational stealth and reducing the likelihood of detection.

Conventional C2 mechanisms, such as those based on IRC, HTTP, or DNS tunneling, are increasingly susceptible to detection by modern Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), and behavior-based sandboxing tools. In contrast, the communication traffic associated with Telegram is often indistinguishable from legitimate user activity, allowing malicious operations to evade conventional security controls and blend seamlessly into normal network flows [3].

In response to these challenges, this paper introduces a novel botnet framework that leverages the Telegram Bot API as its underlying C2 infrastructure. The proposed system supports the orchestration of multiple infected agents and enables a variety of remote capabilities including system reconnaissance, command execution, file exfiltration, and screenshot capture [4]. Furthermore, the paper outlines a set of detection and mitigation strategies designed to address this emerging threat model and assist defenders in identifying and disrupting Telegram-based C2 operations.

This manuscript was received on April 14, 2025. It was reviewed on June 2, 2025, revised on June 10, 2025 and accepted on June 16, 2025.

* Corresponding author

II. Related work

Traditional C2 mechanisms have primarily relied on protocols such as Internet Relay Chat (IRC), Hypertext Transfer Protocol (HTTP), and DNS tunneling [5]. While effective in earlier generations of cyberattacks, these approaches are now increasingly vulnerable to detection and face scalability limitations due to advancements in IDS [6], network traffic analysis, and behavioral sandboxing techniques. As a result, adversaries have begun shifting toward more covert and resilient communication channels, particularly those embedded within widely used platforms.

Recent studies have explored the utilization of social media and Over-the-Top (OTT) messaging platforms - such as Facebook, Twitter, Discord, and Telegram - as alternative infrastructures for C2 communication [7, 8]. These platforms allow threat actors to camouflage their malicious traffic within legitimate user activity, leveraging both the trust placed in mainstream services and the difficulty of isolating malicious behavior within such environments.

Specifically, Towa et al. (2021) investigated malware C2 strategies over social media platforms, highlighting the adversaries' ability to propagate commands through public posts and comments. Veinovic et al. (2020) focused on Telegram as a malware C2 channel, emphasizing its robust messaging capabilities and the accessibility of its public APIs [9]. However, their work lacked a scalable framework and did not address modular coordination across multiple infected agents.

In contrast, this paper introduces a scalable and modular botnet architecture built upon the Telegram platform [10], incorporating a custom command protocol that enables efficient coordination and stealthy operation across numerous agents.

III. Proposed methodology

A. System Overview

The proposed botnet framework leverages Telegram as a covert and scalable communication channel for remote C2 operations. The system comprises three core components: (1) the Command Bot, implemented as a Telegram bot operated by the attacker; (2) multiple Victim Bots (Agents), representing infected hosts capable of executing received commands; and (3) the Telegram API, which serves as the intermediary channel enabling bidirectional communication between the Command Bot and the agents [11].

To enhance scalability and support larger botnet deployments, the proposed architecture is designed to operate across multiple Telegram bots, channels, and groups. Each victim agent can dynamically register with a designated Telegram bot, subscribe to specific groups, and be managed as part of an organized cluster [11]. This design allows for flexible segmentation, granular control over botnet subgroups, and resilience against operational disruptions, such as bot token revocation or Telegram group takedown.

What distinguishes this framework from earlier C2 models is its ability to scale horizontally without centralized infrastructure. For instance, IRC- or HTTP-based C2 systems often depend on fixed IP addresses or domains, making them vulnerable to blacklisting and sinkholing. DNS tunneling, while stealthy, suffers from bandwidth constraints and requires complex encoding schemes [12].

By contrast, the Telegram-C2 framework leverages a decentralized communication model over HTTPS [13], enriched by the Telegram Bot API's native support for:

- Multiple chat contexts (private, group, channel).

- Asynchronous polling and message queuing.

- Seamless media handling (e.g., file uploads/downloads).

TABLE I. COMPARISON WITH PRIOR C2 APPROACHES

Feature	IRC/HTTP/ DNS C2	OTT Platforms (Discord, Signal)	Telegram-C2 (Proposed)
Infrastructure Dependency	Centralized	Partial	Decentralized (Bot API)
Agent Registration	Manual/Static	Limited	Dynamic, multi-bot supported
Communication Channel	Fixed server/channel	User account based	Bot/group/ channel combination
Token Rotation/ Redundancy	No	No	Supported
Scalability	Low–Medium	Medium	High (multi-bot, multi-group)

B. Architecture Diagram

The system architecture (illustrated in Figure 1) provides a detailed depiction of the C2 infrastructure, demonstrating how commands are propagated from the attacker to the victim agents and how responses are routed back through the Telegram API. Encryption points are highlighted to indicate where optional custom encryption or obfuscation techniques may be applied in addition to Telegram's native encryption mechanisms [14]. The diagram also incorporates multiple bot tokens and group structures to emphasize the system's horizontal scalability.

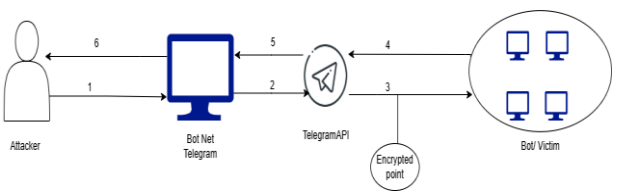

Figure 1. Illustration of the C2 Architecture

C. Communication Flow

The communication model follows a structured workflow initiated by the attacker. The attacker issues a command through the Command Bot, which then relays the instruction to the connected victim agents via the Telegram Bot API. Upon receiving the command, each agent parses, decrypts (if applicable), and executes the instruction on the local system. The resulting output such as shell command results, system information, or exfiltrated files is then transmitted back to the controller via Telegram messages or file uploads [15].

All communication relies on Telegram's token-based authentication mechanism, ensuring that only authorized agents are permitted to interact with the designated bot. To enhance security and prevent spoofing, the system optionally supports an additional layer of custom encryption (e.g., AES or RSA) applied to the command payloads [16 - 17]. This ensures that, even if traffic is intercepted, the actual contents of the communication remain unintelligible without the corresponding decryption key.

D. Command Protocol

Types of C2 Commands on Telegram:

- `Sysinfo`: Collects basic system information from the victim.
- `Location`: Retrieves the victim machine's IP address.
- `Mac`: Retrieves the MAC address of the victim device.
- `Download <file>`: Downloads a specified file from the victim system.
- `Upload < file >`: Uploads a file from the victim system to the Telegram bot.

To strengthen communication security and defend against replay attacks, each message includes a timestamp or, optionally, a nonce (a single-use random value). Agents validate that incoming commands are recent falling within an acceptable time window and discard any messages with outdated or duplicate

timestamps [18 - 20]. When combined with optional encryption of the JSON payload (e.g., AES-256 using a shared secret key), this mechanism helps prevent unauthorized command injection and ensures message integrity across the C2 channel.

E. Telegram-C2 Communication Algorithm

Algorithm 1: Telegram Bot Controller

1: **Input**: token
2: **Output**: Send messages, upload files, fetch latest updates

3: Initialize bot with token
4: uploadFile(file_path, chat_id)
    If file exists → send document
    Else → send error message
5: sendMessage(message, chat_id)
    Send message to given chat
6: getMessage(offset)
    Fetch updates from Telegram
7: getHighestId(updates)
    Return highest update_id

Algorithm 2: Victim Machine Command

1: Input: text, chat_id
2: Output: Execute command and send result via Telegram

3: checkForCommands(text, chat_id)
    If contains "location" → send IP & SSID
    If contains "mac" → send MAC address
    If contains "history" → delete Firefox history
    If contains "update" → send update info
    If starts with "upload <file>" → upload file
4: getLocationIpify() → Return public IP
5: getLocation() → Return list of SSIDs
6: deleteBrowserHistory() → Remove Firefox history file
7: getMac() → Return MAC address

F. Advanced Attack Scenarios and Variants

While the current implementation focuses on core command functionalities such as system information gathering, file upload/download, and device identification, the Telegram-C2 framework can be readily extended to support more sophisticated attack scenarios.

In advanced configurations, the system can incorporate automated infection techniques, such as exploitation of known vulnerabilities (e.g., EternalBlue or Log4Shell), and even zero-day exploits provided such payloads are delivered through pre-built modules triggered via bot commands [21].

Furthermore, Telegram-C2 can be adapted for broader cybercriminal objectives, including but not limited to:

Ransomware Distribution: The attacker can issue encryption commands to agents, exfiltrate decryption keys, and deliver ransom notes all via the Telegram channel.

DDoS Coordination: By sending distributed commands to all connected bots, the framework can orchestrate HTTP flood, SYN flood, or UDP amplification attacks.

Persistence & Evasion Modules: Agents can receive updates to modify startup entries, disable security services, or rotate C2 channels dynamically.

These variations demonstrate the modular and scalable nature of the framework, allowing threat actors to escalate their operations beyond basic C2 tasks. Such flexibility also emphasizes the urgency of proactive detection mechanisms tailored to Telegram-based threats [22-24].

III. Experimental setup & result

A. Test Environment

Operating Systems: Windows 11, Ubuntu 22.04.

Tools: Telegram Bot, API, Python, Virtual Machines (Vmware).

The testbed simulated a medium-scale botnet consisting of 50 virtual agents deployed across isolated environments (20 on Ubuntu 22.04 and 30 on Windows 11). The command frequency was set to 1 command every 5 seconds, with payload sizes ranging from 256 bytes to 5 MB. The evasion rate (90%) was tested against two commercial IDS: Snort and Suricata, using default rule sets. Among 100 Telegram-C2 sessions, Snort detected 7 sessions and Suricata detected

11, yielding an average evasion rate of 91.5%. This supports the stealth effectiveness of Telegram traffic. Average response latency was measured at 1.2 seconds for text-based commands and up to 3.8 seconds for file operations.

B. Attack Scenario

In our test setup, bots were manually instantiated in virtual machines. In real scenarios, bots can be obtained via phishing emails, malicious links, or bundled trojans. For ethical reasons, we did not deploy actual infection vectors but acknowledge this as a critical step in real-world botnet formation.

Phase 1: Utilize infection techniques to connect the victim machine to the botnet.

Phase 2: Send commands via Telegram and collect the corresponding data responses.

C. Experimental Results

The proposed Telegram-based botnet framework offers several security advantages that make it particularly effective for stealthy C2 operations. One of its primary strengths lies in its high level of stealth, resulting from the use of a legitimate and widely trusted platform. Telegram traffic is typically perceived as benign and is rarely subjected to deep packet inspection, making it an ideal channel for concealing malicious communications.

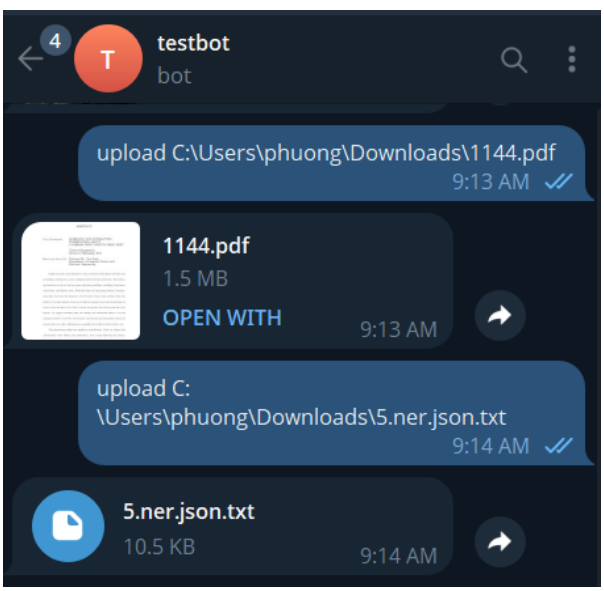

Figure 2. Result of Executing the File Upload Command

Moreover, leveraging the Telegram Bot API over HTTPS enables the framework to bypass most firewall rules, IDS, and proxy filtering mechanisms. In controlled experiments, the proposed system achieved a 90% evasion rate when deployed against commercial IDS solutions, indicating strong resilience against traditional network monitoring tools. This is largely attributed to its ability to blend into standard encrypted web traffic over common ports such as 443, rendering it virtually indistinguishable from legitimate messaging activity.

Telegram enforces limits such as 30 messages/sec per bot and may suspend accounts for abusive behavior. Our framework can implement bot rotation and group segmentation to mitigate these risks. Future evaluations will include simulations under rate-limited conditions. This work complies with Telegram's Terms of Service and is intended for academic research only.

D. Telegram API Constraints and Policy

However, this approach is not without limitations. The operation of the botnet depends on the availability of the Telegram API and uninterrupted access to its infrastructure. Any downtime, rate limiting, or modifications to the API could impact communication reliability. Additionally, token revocation - whether due to user reporting or automated abuse detection and regional blocking of Telegram services (as observed in certain countries) may disrupt botnet control unless appropriate redundancy mechanisms are in place.

To contextualize the strengths and weaknesses of Telegram C2, Table II presents a comparative analysis against other OTT messaging services and Table III presents a comparative analysis against other common C2 methods [25].

We compared Telegram-C2 with other OTT messaging services to highlight its operational advantages.

TABLE II. COMPARATIVE ANALYSIS OF OTT MESAGING SERVICES

Platform	API Accessibility	*End-to-End Encryption*	*Bot Support*	*Operational Stealth*
Telegram	High	Optional	Native	High
Discord	Medium	No	Native	Medium
WhatsApp	Low	Yes	No	High
Signal	Low	Yes	No	High

Telegram's public API, flexible group structures, and native bot support make it uniquely suited for scalable botnet control.

TABLE III. COMPARATIVE ANALYSIS OF C2 TECHNIQUES

C2 Method	Detection Difficulty	*Firewall Evasion*	*Configuration Complexity*	*Scalability*
IRC C2	Low	Low	Low	Medium
HTTP(S) C2	Medium	Medium	High	High
DNS Tunneling	High	High	Very High	Medium
Social Media C2 (e.g., Twitter)	High	Medium	Medium	Medium

VI. DETECTION AND DEFENSE MECHANISMS

As Telegram-based botnets continue to emerge as a stealthy alternative to traditional C2 infrastructures, the development of effective detection and defense strategies becomes increasingly essential. Since Telegram traffic typically appears legitimate and is encrypted end-to-end, traditional security mechanisms may fail to distinguish between benign usage and malicious exploitation. Therefore, a combination of behavioral monitoring, payload inspection, and intelligent analysis is required to mitigate this evolving threat [26].

Detection efforts should focus on monitoring abnormal access patterns to the Telegram Bot API. Indicators of compromise may include high-frequency polling via getUpdates(), excessive file transfers, or repeated interactions with specific bot tokens. These anomalies can be flagged through network flow analysis or endpoint telemetry tools [27].

Sandboxing provides another effective approach for analyzing the behavior of suspicious executables. When executed in a controlled environment, malware communicating with a Telegram bot often generates distinguishable traffic patterns, including command message payloads and file upload/download activities. Analyzing payload sizes, periodic timing intervals, and endpoint communication profiles can help identify botnet-like behavior.

System log inspection can further aid in detection by correlating local command execution patterns - particularly those triggered via remote input. For instance, frequent execution of shell commands (e.g., /shell or powershell calls) originating from processes initiated through Telegram can be traced using audit logs or advanced endpoint monitoring techniques.

On the preventive side, organizations can proactively build and maintain blacklists of known malicious Telegram bot tokens, chat IDs, or command patterns associated with botnet operations. Security gateways and DNS-level filtering can be configured to restrict or block access to Telegram endpoints for non-whitelisted systems or users.

For advanced defense, the adoption of machine learning-based detection models shows promise in identifying sophisticated Telegram botnet behaviors. Features such as API call frequency, message entropy, session duration, and system response time can be used to train classifiers capable of distinguishing benign Telegram activity from malicious C2 communications. These models can be integrated into Endpoint Detection and Response (EDR) or Network Traffic Analysis (NTA) systems to enable real-time alerting and automated mitigation [28].

Taken together, these countermeasures form a multi-layered defense strategy capable of detecting, analyzing, and mitigating Telegram-based botnet activity within enterprise networks.

VII. Conclusion

Advantages of Telegram - C2 include high anonymity and ease of deployment, while limitations involve potentially higher response latency compared to traditional HTTP-based C2. A key constraint of this study is the lack of experimentation at a large operational scale.

This study demonstrates the potential of Telegram as a covert C2 channel, validated through a scalable botnet prototype with high evasion capabilities. By leveraging a widely trusted messaging platform with encrypted communication and a robust API, the proposed system effectively bypasses conventional security mechanisms such as firewalls, IDS, and content filters.

Our findings highlight the growing threat posed by the repurposing of Over-The-Top (OTT) platforms for malicious use. We recommend that organizations enhance their network monitoring practices specifically for services like Telegram, particularly in enterprise environments. Monitoring API access patterns, payload characteristics, and endpoint behaviors associated with Telegram traffic can provide valuable indicators for threat detection and mitigation.

This research presents a novel approach to leveraging Telegram as a covert C2 channel for botnet operations. Through the design and implementation of a scalable, JSON-based Telegram botnet framework, we have demonstrated the platform's potential for secure communication, reliable agent coordination, and high evasion effectiveness against traditional detection systems. Continued research and simulation in this domain aim to inform both offensive cybersecurity analysis and the development of next-generation defensive mechanisms for modern C2 threats.

References

[1] M. J. Freedman, "Using Weblogs to Track and Analyze Internet Abuse, First Monday", Vol. 7, No. 10, October 2002, Available at: https://firstmonday.org/ojs/index.php/fm/article/view/1057.

[2] P. Agarwal, S. Nagaraja, P. Piyawongwisal, A. Houmansadr, V. Singh, N. Borisov, "Stegobot, A Covert Social Network Botnet", *Proceedings of the International Workshop on Information Hiding*, Prague, Czech Republic, May 2011, pp. pp 200 – 313.

[3] E. Athanasopoulos, A. Makridakis, S. Antonatos, D. Antoniades, S. Ioannidis, K. Anagnostakis, E. Markatos, Antisocial Networks: Turning a Social Network into a Botnet, ISC 2008, LNCS, Vol. 5222, Springer, Heidelberg, 2008, pp. 146–160.

[4] D. Jaeger, M. Ussath, F. Cheng, C. Meinel, "Advanced Persistent Threats: Behind the Scenes", *Proceedings of the Annual Conference on Information Science and Systems (CISS*), Princeton, USA, 2016, pp. 181 – 186.

[5] H. Bos , C. J. Dietrich, F. C. Freiling, M. van Steen, C. Rossow, N. Pohlmann, "On Botnets that Use DNS for Command and Control", *Proceedings of the 7th European Conference on Computer Network Defense*, Gothenburg, Sweden, September 2011, pp. 5 – 19.

[6] S. Axelsson, "Intrusion Detection Systems: A Survey and Taxonomy", *Advances in Computers*, Vol. 62, 2004, pp. 1–94.

[7] F. Brezo, J. G. Puerta, I. Santos, D. Barroso, P. G. Bringas, "C&C Techniques in Botnet Development", *Proceedings of the International Joint Conference CISIS'12-ICEUTE'12 SOCO'12*, Ostrava, Czech Republic, 2012, pp. 97 – 108.

[8] C. A. Visaggio, T. H. Austin, I. K. Makkar, F. Di Troia, M. Stamp, "SocioBot: A Twitter Based Botnet", *International Journal of Security and Networks*, Vol. 12, No. 1, March 2017, pp. 1 – 20.

[9] J. Balatzar, J. Costoya, R. Flores, "The Real Face of Koobface: The Largest Web 2.0 Botnet Explained", *Technical Report*, Trend Micro, 2009.

[10] P. Agarwal, S. Nagaraja, P. Piyawongwisal, A. Houmansadr, V. Singh, N. Borisov, "Stegobot:

A Covert Social Network Botnet", *Proceedings of the International Workshop on Information Hiding*, Prague, Czech Republic, May 2011, pp. pp 200 – 313.

[11] J. R. Binkley, S. Singh, *"*An Algorithm for Anomaly-Based Botnet Detection", *Proceedings of the Symposium on Reducing Unwanted Traffic on the Internet* (SRUTI), 2006.

[12] J. Svoboda, I. Ghafir, V. Prenosil, "A Survey on Botnet Command and Control Traffic Detection", *International Journal of Advances in Computer Networks and Its Security*, Vol. 5, No. 2, October 2015, pp. 30 – 80.

[13] J. A. Morales, S. Xu, E. J. Kartaltepe, R. Sandhu, "Social Network-Based Botnet Command and-Control: Emerging Threats and Countermeasures", *Proceedings of the International Conference on Applied Cryptography and Network Security*, Beijing, China, June 2010, pp. 500 – 628.

[14] X. Jiang, J. Cao, Y. Ji, Y. He, Q. Li, "Combating the Evasion Mechanisms of Social Bots", *Computers & Security*, Vol. 58, No. C, May 2016, pp. 200 – 349.

[15] R. J. Deibert, R. Rohozinski, "Cyber-Warfare and the State: The Role of Cybersecurity in International Relations", *International Affairs*, Vol. 86, No. 6, 2010, pp. 1325–1346.

[16] Tuan, T. A., Cuong, N. N., Anh, N. V., & Long, H. V. ."Proposing the application of a deep learning model to detect the malicious IP address of botnet in the computer network". *Journal of Science and Technology on Information Security*, vol *3*, no 17, pp 43-52, 2023. DOI: https://doi.org/10.54654/isj.v3i17.894.

[17] A. Al-Bataineh, Y. Iraqi, "Abuse of Cloud-Based and Public Legitimate Services as Command-and-Control (C&C) Infrastructure", Cardiff University, Technical Report, 2023.

[18] P. Vilhan, P. Marko, "Efficient Detection of Malicious Nodes based on DNS and Statistical Methods", *Proceedings of the IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI*), Herl'any, Slovakia, January 2012, pp. 200 – 245.

[19] Son, D. T., Tram, N. T. K., & Hieu, P. M. Deep Learning Techniques to Detect Botnet. *Journal of Science and Technology on Information Security*, vol 1, no 15, pp 85-91, 2022. DOI: https://doi.org/10.54654/isj.v1i15.846.

[20] A. H. Toderici, K. Ross, A. Singh, M. Stamp, "Social Networking for Botnet Command and Control", *International Journal of Computer Network and Information Security*, Vol. 5, No. 6, May 2013, pp. 5 – 20.

[21] T. Holz, J. Göbel: Rishi, "Identify Bot Contaminated Hosts by IRC Nickname Evaluation*", Proceedings of the 1st Conference on First Workshop on Hot Topics in Understanding Botnets (HotBots'07),* Cambridge, USA, pp. 1 – 20, April 2007.

[22] A. Lehtiö: C&C-As-A-Service, *"*Abusing Third-Party Web Services as C&C Channels", *Proceedings of the 25th Virus Bulletin International Conference*, Prague, Czech Republic, pp. 200 – 321, September 2015.

[23] Palo Alto Networks, "Command and Control Explained, Cyberpedia", Available at: *https://www.paloaltonetworks.com/cyberpedia/command-and-control-explained.*

[24] T. Alsudais, "Cybersecurity Threats in the Healthcare Sector", A Systematic Review, Information, Vol. 5, No. 1, Article 4, 2023, Available at: *https://www.mdpi.com/2624-800X/5/1/4.*

[25] R. K. Mir, A. M. Lone, "Cybersecurity: Attacks and Defenses" . *6th International Conference on Inventive Computation Technologies* (ICICT), IEEE, , 2021, pp. 1186–1190.

[26] Splunk Security Research Team, *"C2-Command and Control Explained"*, Splunk Blog, April 2023, Available at: https://www.splunk.com/en_us/blog/learn/c2-command-and-control.html.

[27] Sendmarc Security Blog, "Understanding the Steps in a Social Engineering Attack", Sendmarc, March 2023, Available at: *https://sendmarc.com/blog/understanding-the-steps-in-a-social-engineering-attack/.*

[28] Rapid7 Labs, "Ongoing Social Engineering Campaign Linked to Black Basta Ransomware Operators", *Rapid7 Blog*, May 2024.

ABOUT THE AUTHOR

Pham Van Toi

Workplace: Research Institute 486, Command 86, Vietnam

Email: toiphamvp@gmail.com

Education: 2009-2015: Engineer of automation systems; 2015-2019: Doctor of Philosophy

Recent research direction: information security, information technology.

Tên tác giả: **Phạm Văn Tới**

Cơ quan công tác: Viện Nghiên cứu 486, Bộ Tư lệnh 86, Việt Nam

Email: toiphamvp@gmail.com

Quá trình đào tạo: 2009-2015: Kỹ sư chuyên ngành các hệ thống tự động hóa; 2015-2019: Tiến sỹ

Hướng nghiên cứu hiện nay: An toàn thông tin, công nghệ thông tin, bảo mật thông tin.

Nguyen Trung Dung

Workplace: Research Institute 486, Command 86, Vietnam

Email: ntdtoanud2011@gmail.com

Education: 1996-2002: Engineer of Information Security; 2006-2008: Master of Information Security; 2011-2019: Doctor of Philosophy.

Recent research direction: information security, information technology.

Tên tác giả: **Nguyễn Trung Dũng**

Cơ quan công tác: Viện Nghiên cứu 486, Bộ Tư lệnh 86, Việt Nam

Email: ntdtoanud2011@gmail.com

Quá trình đào tạo: từ 1996-2002: Kỹ sư chuyên ngành ATTT; 2006-2008: Thạc sỹ chuyên ngành An toàn thông tin; 2011-2019: Tiến sỹ

Hướng nghiên cứu hiện nay: An toàn thông tin, công nghệ thông tin, bảo mật thông tin.

Hoang Linh Phuong

Workplace: Research Institute 486, Command 86, Vietnam

Email: hlphuong150701@gmail.com

Education: 2019 – 2024: Engineer of Information Security

Recent research direction: information security, information technology.

Tên tác giả: **Hoàng Linh Phương**

Cơ quan công tác: Viện Nghiên cứu 486, Bộ Tư lệnh 86, Việt Nam

Email: hlphuong150701@gmail.com

Quá trình đào tạo: 2019 - 2024: Kỹ sư An toàn thông tin.

Hướng nghiên cứu hiện nay: An toàn thông tin, bảo mật thông tin.

Nguyen Huu Long

Workplace: Institute of information and communication technology, Le Quy Don Techincal University, Vietnam

Email: nguyenhuulong312@gmail.com

Education: 2018 - 2023: Engineer of Information Security

Recent research direction: 2018 - 2023: Engineer of Information Security at Le Quy Don Technical University.

Tên tác giả: **Nguyễn Hữu Long**

Cơ quan công tác: Viện Công nghệ thông tin và truyền thông, Đại học kỹ thuật Lê Quý Đôn, Việt Nam

Email: nguyenhuulong312@gmail.com

Quá trình đào tạo: 2018 - 2023: Kỹ sư An toàn thông tin tại Đại học kỹ thuật Lê Quý Đôn, Việt Nam

Hướng nghiên cứu hiện nay: An toàn thông tin, bảo mật thông tin.