ITEJ
Information Technology Engineering Journals
eISSN : 2548-2157

Url : https://syekhnurjati.ac.id/journal/index.php/itej
Email : itej@syekhnurjati.ac.id

Sharing SSH Threat Intelligence across Multiple Servers using WebSocket and Fail2Ban

Aristo Tely
Telecommunications Engineering
State Polytechnic of Sriwijaya
telyaristo@gmail.com

Aryanti Aryanti*
Telecommunications Engineering
State Polytechnic of Sriwijaya
aryanti@polsri.ac.id

Sopian Soim
Telecommunications Engineering
State Polytechnic of Sriwijaya
sopiansoim@gmail.com

Abstract—This study presents a lightweight prototype designed to improve SSH brute-force defense by enabling collaborative IP blocking across multiple servers. The system integrates Fail2Ban with WebSocket to distribute banned IP addresses in real-time among trusted nodes eliminating the need for centralized infrastructure. The experiment was conducted on 3 virtual private servers (VPS), where one acted as the WebSocket server and the others as clients equipped with Fail2Ban. When an SSH brute-force attack is detected, the source IP is automatically shared across the network and blocked on all connected nodes. A qualitative observational approach was used to evaluate the system's feasibility. Log data from the clients and server was analyzed to confirm the accuracy and consistency of IP synchronization. The results showed that banned IPs were propagated and enforced on all nodes within seconds of detection. These findings demonstrate the potential for decentralized, lightweight collaboration among SSH servers to enhance security without introducing complex infrastructure or external dependencies.

Keywords—Fail2Ban, WebSocket, SSH Security, Intrusion Prevention System, Threat Intelligence Sharing.

I. Introduction

Secure Shell (SSH) is widely regarded as the backbone protocol for secure remote access and server management. However, despite its cryptographic robustness, it remains a primary target for cyber attackers, particularly through brute-force login attempts. These attacks involve repeated attempts to guess credentials and gain unauthorized access, often executed using distributed botnets, which makes them harder to detect and mitigate[1][2]. Studies have shown a consistent rise in SSH brute-force attempts, particularly across cloud-based and virtualized infrastructures[3][4].

To combat this, tools like Fail2Ban have been extensively adopted due to their lightweight and host-level protection capabilities. Fail2Ban works by monitoring authentication logs and banning IP addresses that exceed a certain threshold of failed login attempts[5][6]. Its effectiveness in preventing localized brute-force attacks is well-documented[7][8]. However, it inherently operates in isolation, each server only protects itself, and no intelligence is shared with other machines[9][10]. This creates a critical vulnerability: if Server A blocks a malicious IP, Server B remains unaware and potentially exposed to the same threat.

This paper proposes a lightweight prototype that leverages WebSocket and Fail2Ban to enable real-time sharing of brute-force threat intelligence across SSH servers. Rather than relying on centralized databases or community consensus, the system facilitates immediate communication among trusted nodes using persistent WebSocket connections.

When a malicious IP is detected and banned on one server, it is instantly propagated to all others in the network. This collaborative model aims to shorten response time, reduce duplicated attacks, and strengthen collective resilience against SSH brute-force threats.

Unlike complex distributed IDS solutions, this prototype emphasizes simplicity, flexibility, and ease of integration, making it suitable for medium-scale infrastructures or intra-organizational server environments. The focus of this study is not on quantitative benchmarking, but rather a qualitative exploration of how decentralized communication enhances proactive defense in SSH systems.

II. RELATED WORKS

Several approaches have been proposed to overcome the limitations of host-based intrusion prevention. Early works explored integrating Fail2Ban with centralized databases to propagate blocklists across multiple servers. For instance, T. Kumar et al. demonstrated a distributed Fail2Ban deployment in HPC clusters where all nodes synchronized their blocked IPs through a shared SQL backend[11]. While effective within small-scale environments, centralized approaches suffer from single points of failure, synchronization bottlenecks, and maintenance overhead[12][13].

Community-based platforms like CrowdSec aim to expand threat intelligence sharing at a global scale. Each CrowdSec agent contributes to and receives malicious IP data from a centralized consensus engine, enabling preemptive IP blocking across participating system[14][15]. Although promising, this model depends heavily on external infrastructure and trust management, which may not suit more isolated or sensitive deployments[16][17].

Decentralized alternatives have gained attention for their ability to reduce reliance on third parties. Distributed Intrusion Detection Systems (DIDS) and cooperative frameworks have been proposed using peer-to-peer communication among nodes[18][19]. These systems leverage local observations across multiple machines to generate collective decisions and mitigate threats earlier. Studies on multi-agent systems in cybersecurity further reinforce the advantages of distributed coordination, adaptive behavior, and autonomous threat response[20][21].

In parallel, mobile agents have emerged as a technique for dynamic and flexible intrusion detection. These agents can migrate between hosts, carrying detection logic and contextual awareness[22]. However, they often introduce additional attack surfaces and complexity, which may not be suitable for lightweight environments[23]. To enhance their resilience, methods such as dummy-based decoy systems have been proposed to protect mobile agents from hostile destinations[24].

Communication plays a key role in distributed defense. Modern architectures have explored using WebSocket as a persistent and real-time communication layer to propagate events and security alerts between distributed nodes[25]. Unlike polling mechanisms, WebSocket enables bidirectional, low-latency messaging that suits collaborative defense use cases. Applications of WebSocket have been documented in cloud monitoring, DDoS mitigation, and real-time anomaly detection[26].

III. METHOD

This study uses a qualitative observational approach to evaluate the behavior of a prototype system designed to distribute SSH brute-force attack intelligence between multiple servers. The prototype integrates Fail2Ban with WebSocket to enable real-time synchronization of IP block actions among nodes. The method consists of 3 main stages: system design, implementation, and observation.

A. System Design

The system consists of four VPS instances: one designated as the central WebSocket server and 3 as SSH servers equipped with Fail2Ban and WebSocket client agents. The WebSocket server receives block/unblock events from any client and broadcasts them to all other connected clients. Each client is equipped with Fail2Ban to detect brute-force attempts, a script trigger to send blocked IPs to the WebSocket server, and a listener agent to receive and apply blocking instructions from the server.

The overall architecture is illustrated in Figure 1. Each component communicates via persistent WebSocket channels, and no centralized database is used. Clients independently enforce bans using local firewall rules (iptables).

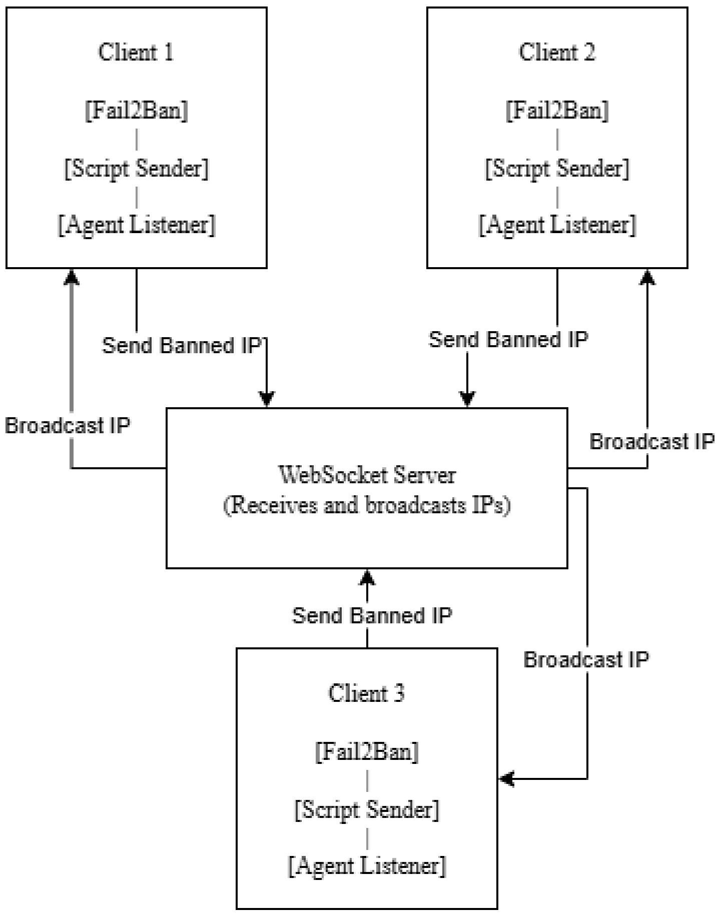
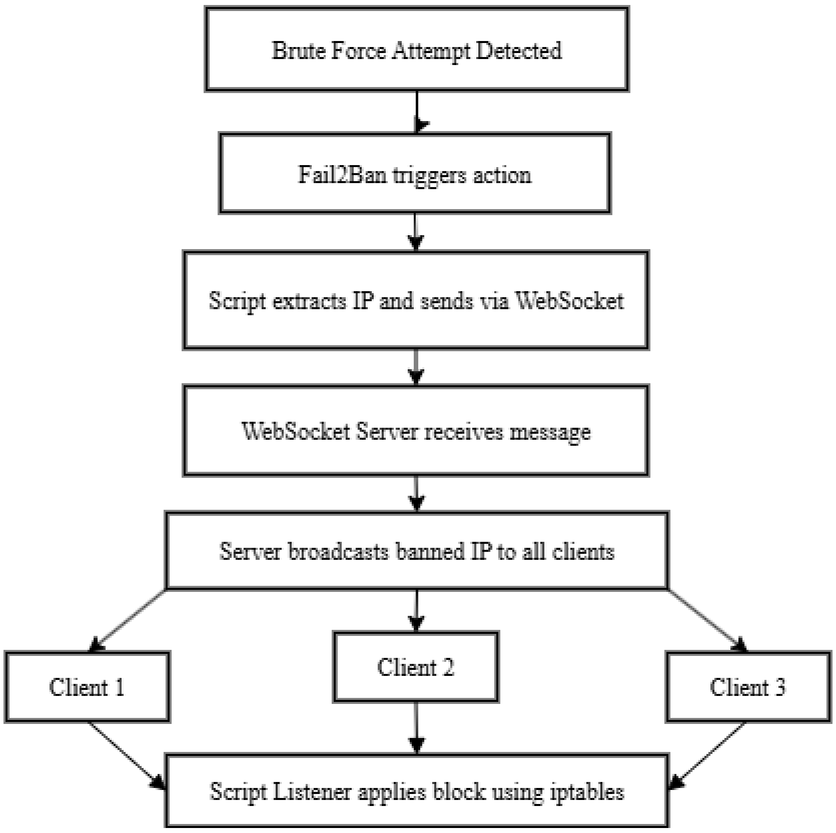


Figure 1. System Architecture

B. Implementation Flow

The flow of data and actions within the system is illustrated in Figure 2. The communication begins when Fail2Ban detects a sequence of failed login attempts that exceed the defined threshold. This triggers a predefined action script responsible for extracting the offending IP address and sending it to the WebSocket server using a secure, persistent connection. Upon receipt, the WebSocket server immediately broadcasts this IP to all connected client nodes using a push mechanism, ensuring that the message reaches each recipient in near real-time. On the client side, a lightweight listener agent receives the broadcasted message and invokes a local script to apply a firewall rule that blocks the malicious IP using iptables. This sequence enables rapid propagation of threat intelligence across all SSH nodes without requiring any centralized database or polling cycle.


Brute Force Attempt Detected
Fail2Ban triggers action
Script extracts IP and sends via WebSocket
WebSocket Server receives message
Server broadcasts banned IP to all clients
Client 1
Client 2
Client 3
Script Listener applies block using iptables
Figure 2. Data Flow Diagram

Unblocking events are similarly propagated using scheduled jobs or manual triggers. The system is designed to operate asynchronously and without requiring feedback from peer nodes. The response time is calculated as shown in:

$$RT = T(Block) - T(Detect) \qquad (1)$$

where RT denotes the response time, and T refers to the corresponding timestamps of the block and detection events.

C. Observation Procedure

The experiment was conducted by simulating brute-force SSH login attempts on one of the client servers, specifically Client 1, to observe how the prototype system responds across all participating nodes. The simulation involved repeated failed login actions using automated tools designed to trigger the Fail2Ban threshold within seconds. Once triggered, Fail2Ban executed a predefined shell script that relayed the malicious IP address to the WebSocket server, which then broadcast the information to the remaining clients.

To verify the system’s reaction, detailed log files were collected from all clients and the WebSocket server. These included:

a. Raw action logs documenting all triggered events, such as BLOCK, UNBLOCK, and SCHEDULE_UNBLOCK, which allowed researchers to trace each step of the response lifecycle.
b. Structured CSV logs containing timestamps, offending IP addresses, and associated actions, offering precise measurements of response intervals between detection and enforcement.

A representative sample from Client 1's logs on July 2, 2025, recorded the following sequence as summarized below:

- 11:52:22: Detected brute-force from IP 128.124.20.205, executed BLOCK.
- 11:52:22: SCHEDULE_UNBLOCK created for 120 minutes.
- 11:52:35: IP was unblocked automatically.
- 11:54:05: IP 186.117.149.128 was blocked.

These observations confirmed that the system responded consistently and as expected under real-world conditions. All events were logged accurately and showed synchronized actions across clients following the WebSocket broadcasts. The experimental deployment and its network topology are illustrated in Figure 3, which shows the layout of the VPS nodes and their communication relationships. This visual helps to contextualize the distributed nature of the system and supports the qualitative insights drawn from the observation. It is important to note that this study did not involve stress testing or analysis of resource consumption; rather, it focused purely on validating the system's functional feasibility and accuracy in propagating block actions across multiple nodes.

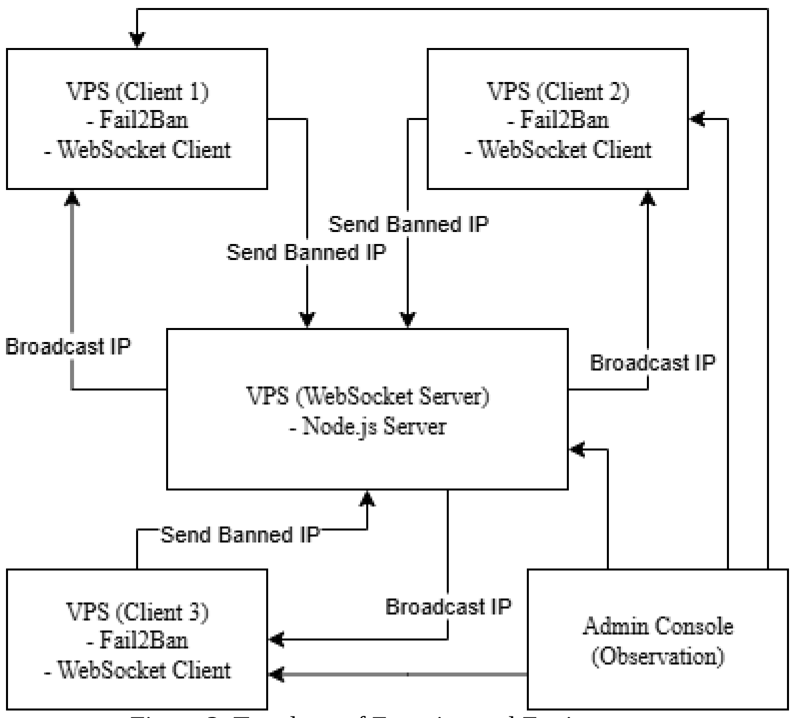


Figure 3. Topology of Experimental Environment

IV. Result and Discussion

The experiment was conducted using 3 client nodes and one WebSocket server. Each client was equipped with Fail2Ban and a WebSocket-based communication agent, while the server acted as a hub to relay blocked IP addresses. The goal of this observation was to confirm whether banned IPs could be reliably shared across nodes and applied in real-time without introducing significant overhead or synchronization issues.

One of the key observations was the handling of the IP address 128.124.20.205. At 11:52:22, Client 1 detected a brute-force attempt and issued a BLOCK action through Fail2Ban. Simultaneously, a scheduled unblock event was created, set to remove the block after 120 minutes. However, at 11:52:35, a premature UNBLOCK action was registered. This sequence showed that the prototype correctly executed and logged each phase of the blocking lifecycle, including the automation of unblocking and handling of overlapping timers.

Another example was observed at 11:54:05, when IP 186.117.149.128 was detected and blocked by Client 1. Although the observation scope was limited, these events were successfully relayed to the WebSocket server and forwarded to the other clients. The log entries showed that within approximately 1–2 seconds of the original BLOCK action, the same IP address appeared in the logs of other nodes. This indicates that the synchronization mechanism worked consistently and met the expected delay threshold for lightweight communication over WebSocket. These observations are summarized in Table 1, which illustrates the sequence of blocking actions across the participating nodes.

Table 1. Sample of Synchronized IP Blocking Events

Timestamp	Node	Action	IP Address	Description
11:52:22	Client 1	BLOCK	128.124.20.205	Initial detection
11:52:22	Client 1	SCHEDULE_UNBLOCK	128.124.20.205	Timed unblock set to 120 minutes
11:52:35	Client 1	UNBLOCK	128.124.20.205	Early unblocking observed
11:52:23	Client 2	BLOCK	128.124.20.205	Received from server broadcast
11:52:23	Client 3	BLOCK	128.124.20.205	Received from server broadcast
11:54:05	Client 1	BLOCK	186.117.149.128	New brute-force source detected
11:54:06	Client 2	BLOCK	186.117.149.128	Synchronized block applied
11:54:06	Client 3	BLOCK	186.117.149.128	Synchronized block applied

To further validate the behavior of the system, timestamp comparisons were performed between client-side and server-side logs. These comparisons revealed near-identical sequences of actions across nodes, with minimal deviation in timing. Such consistency is especially valuable in multi-server defense systems, where synchronized bans can prevent attackers from rotating between targets.

The overall response time from detection to enforcement across clients remained within an acceptable range for a lightweight implementation. While no precise latency benchmarks were calculated, the qualitative results remain valuable for understanding system behavior. This also demonstrates that real-time collaboration among distributed nodes is achievable without requiring complex orchestration or centralized infrastructure. In particular, the system’s ability to react almost instantaneously across all clients adds a layer of proactive mitigation to SSH security.

Throughout the testing period, no duplicated blocking or missed synchronization was observed, suggesting a high success rate in IP propagation. It also implies that the message delivery over the WebSocket channel remained stable during the entire experiment. This is an important indicator for real-world deployment, as reliability and message integrity are critical in any security-driven communication model.

This study did not yet address edge cases such as system failure conditions, message loss, or race conditions between simultaneous blocking and unblocking events. These aspects are crucial for scaling the system to larger environments and will be addressed in future work. Further analysis involving traffic load simulation, intentional message drops, and node failures would be needed to quantify the system’s resilience and fault tolerance.

These findings, taken together, demonstrate that it is feasible to strengthen SSH server defense through real-time cooperation by utilizing a straightforward WebSocket-based approach. While this prototype is limited to basic IP sharing and observability, it lays the groundwork for more adaptive and scalable implementations in future studies. The potential to extend this system to broader cloud-based environments or integrate with intrusion detection systems (IDS) presents a promising direction for future development.

V. Conclusion

This study presented an exploratory implementation of a lightweight and decentralized system for mitigating SSH brute-force attacks through real-time IP sharing using Fail2Ban and WebSocket. Designed to operate across three server nodes, the system allowed each node to autonomously detect intrusion attempts and distribute banned IPs without requiring centralized infrastructure or third-party intelligence services. The results from log-based observation confirmed that the system consistently propagated and enforced block actions across all clients within a short delay, validating its functional feasibility and synchronization accuracy. Although the evaluation focused solely on functional behavior rather than performance under stress or failure scenarios, the findings demonstrate that lightweight collaboration between SSH servers is both possible and effective. This prototype serves as a foundational step toward developing more adaptive, scalable, and robust distributed defense systems, particularly in resource-constrained environments. Future work will expand on these findings by incorporating dynamic configuration handling, fault tolerance, and integration with broader cloud-based security frameworks to support more complex and real-world deployments.

References

[1] A. Irsheid, A. Murad, M. AlNajdawi, and A. Qusef, "Information security risk management models for cloud hosted systems: A comparative study," *Procedia Comput Sci*, vol. 204, pp. 205–217, 2022, doi: https://doi.org/10.1016/j.procs.2022.08.025.

[2] A. Kumar, I. Budhiraja, D. Garg, S. Garg, B. J. Choi, and M. Alrashoud, "Advanced network security with an integrated trust-based intrusion detection system for routing protocol," *Alexandria Engineering Journal*, vol. 120, pp. 378–390, May 2025, doi: https://doi.org/10.1016/j.aej.2025.01.087.

[3] A. F. Otoom, W. Eleisah, and E. E. Abdallah, "Deep Learning for Accurate Detection of Brute Force attacks on IoT Networks," *Procedia Comput Sci*, vol. 220, pp. 291–298, 2023, doi: https://doi.org/10.1016/j.procs.2023.03.038.

[4] S. Sentanoe and H. P. Reiser, "SSHkex: Leveraging virtual machine introspection for extracting SSH keys and decrypting SSH network traffic," *Forensic Science International: Digital Investigation*, vol. 40, p. 301337, Apr. 2022, doi: https://doi.org/10.1016/j.fsidi.2022.301337.

[5] P. Ajay, B. Nagaraj, R. Arun Kumar, V. Suthana, and M. Ruth Keziah, "DBN-protected material Enhanced intrusion prevention sensor system defends against cyber attacks in the IoT devices," *Measurement: Sensors*, vol. 34, p. 101263, Aug. 2024, doi: https://doi.org/10.1016/j.measen.2024.101263.

[6] A. Allami, T. Nicewarner, K. Goss, A. Kundu, W. Jiang, and D. Lin, "Oblivious and distributed firewall policies for securing firewalls from malicious attacks," *Comput Secur*, vol. 150, p. 104201, Mar. 2025, doi: https://doi.org/10.1016/j.cose.2024.104201.

[7] M. Srinivasan and N. C. Senthilkumar, "Intrusion Detection and Prevention System (IDPS) Model for IIoT Environments Using Hybridized Framework," *IEEE Access*, vol. 13, pp. 26608–26621, 2025, doi: https://doi.org/10.1109/ACCESS.2025.3538461.

[8] R. V. Mendonca *et al.*, "Intrusion Detection System Based on Fast Hierarchical Deep Convolutional Neural Network," *IEEE Access*, vol. 9, pp. 61024–61034, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3074664.

[9] J. Park, J. Kim, B. B. Gupta, and N. Park, "Network Log-Based SSH Brute-Force Attack Detection Model," *Computers, Materials & Continua*, vol. 68, no. 1, pp. 887–901, 2021, doi: https://doi.org/10.32604/cmc.2021.015172.

[10] T. Mohamed Ahmed, “Developing Check-Point Mechanism to Protect Mobile Agent Free-Roaming Against Untrusted Hosts,” *Computers, Materials & Continua*, vol. 72, no. 2, pp. 3849–3862, 2022, doi: https://doi.org/10.32604/cmc.2022.025582.

[11] T. Kumar, P. Sharma, X. Cheng, S. Lalar, S. Kumar, and S. Bansal, “Enhanced Triple Layered Approach for Mitigating Security Risks in Cloud,” *Computers, Materials & Continua*, vol. 83, no. 1, pp. 719–738, 2025, doi: https://doi.org/10.32604/cmc.2025.060836.

[12] E. Seid, O. Popov, and F. Blix, “Evaluation of Asfalia, a Security Attack Event Monitoring Framework,” *Procedia Comput Sci*, vol. 237, pp. 793–802, 2024, doi: https://doi.org/10.1016/j.procs.2024.05.167.

[13] S. Girish Savadatti, K. Srinivasan, and Y.-C. Hu, “A Bibliometric Analysis of Agent-Based Systems in Cybersecurity and Broader Security Domains: Trends and Insights,” *IEEE Access*, vol. 13, pp. 90–119, 2025, doi: https://doi.org/10.1109/ACCESS.2024.3520583.

[14] M. J. Shayegan and A. Damghanian, “A Method for DDoS Attacks Prevention Using SDN and NFV,” *IEEE Access*, vol. 12, pp. 108176–108184, 2024, doi: https://doi.org/10.1109/ACCESS.2024.3438538.

[15] M. A. Elsadig, “Detection of Denial-of-Service Attack in Wireless Sensor Networks: A Lightweight Machine Learning Approach,” *IEEE Access*, vol. 11, pp. 83537–83552, 2023, doi: https://doi.org/10.1109/ACCESS.2023.3303113.

[16] J. Halladay *et al.*, “Detection and Characterization of DDoS Attacks Using Time-Based Features,” *IEEE Access*, vol. 10, pp. 49794–49807, 2022, doi: https://doi.org/10.1109/ACCESS.2022.3173319.

[17] H. Artajaya, Julieta, J. Giancarlos, J. V. Moniaga, and A. Chowanda, “Development of a Secure Web Based Application to Automate Data Synchronization and Processing,” *Procedia Comput Sci*, vol. 245, pp. 1175–1181, 2024, doi: https://doi.org/10.1016/j.procs.2024.10.347.

[18] I. A. Saeed, A. Selamat, M. F. Rohani, O. Krejcar, and J. A. Chaudhry, “A Systematic State-of-the-Art Analysis of Multi-Agent Intrusion Detection,” *IEEE Access*, vol. 8, pp. 180184–180209, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3027463.

[19] J. E. Varghese and B. Muniyal, “An Efficient IDS Framework for DDoS Attacks in SDN Environment,” *IEEE Access*, vol. 9, pp. 69680–69699, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3078065.

[20] M. Nadeem, A. Arshad, S. Riaz, S. S. Band, and A. Mosavi, “Intercept the Cloud Network From Brute Force and DDoS Attacks via Intrusion Detection and Prevention System,” *IEEE Access*, vol. 9, pp. 152300–152309, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3126535.

[21] S. Al Amro, “Securing Internet of Things Devices with Federated Learning: A Privacy-Preserving Approach for Distributed Intrusion Detection,” *Computers, Materials & Continua*, vol. 83, no. 3, pp. 4623–4658, 2025, doi: https://doi.org/10.32604/cmc.2025.063734.

[22] A. M. Alnajim, F. M. Alotaibi, and S. Khan, “Detecting and Mitigating Distributed Denial of Service Attacks in Software-Defined Networking,” *Computers, Materials & Continua*, vol. 83, no. 3, pp. 4515–4535, 2025, doi: https://doi.org/10.32604/cmc.2025.063139.

[23] B. Alluhaybi, M. S. Alrahhal, A. Alzahrani, and V. Thayananthan, “Dummy-Based Approach for Protecting Mobile Agents Against Malicious Destination Machines,” *IEEE Access*, vol. 8, pp. 129320–129337, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3009245.

[24] M. Elgamal, A. Abdel Menaem, M. A. Alotaibi, V. Oboskalov, and A. Elmitwally, “Distributed agents structure for current-only adaptive relaying

scheme reinforced against failures and cyberattacks," *Ain Shams Engineering Journal*, vol. 15, no. 12, p. 103143, Dec. 2024, doi: https://doi.org/10.1016/j.asej.2024.103143.

[25] A. S. Abdelfattah, T. Abdelkader, and E.-S. M. EI-Horbaty, "RAMWS: Reliable approach using middleware and WebSockets in mobile cloud computing," *Ain Shams Engineering Journal*, vol. 11, no. 4, pp. 1083–1092, Dec. 2020, doi: https://doi.org/10.1016/j.asej.2020.04.002.

[26] M. Ouhssini, K. Afdel, M. Akouhar, E. Agherrabi, and A. Abarda, "Advancements in detecting, preventing, and mitigating DDoS attacks in cloud environments: A comprehensive systematic review of state-of-the-art approaches," *Egyptian Informatics Journal*, vol. 27, p. 100517, Sep. 2024, doi: https://doi.org/10.1016/j.eij.2024.100517.