



HTTP Flood Detection for Smart Agriculture Using Nginx and Flask

Kavita Kumavat¹  and Sanjesh Pawale¹ 

Department of Computer Engineering, Vishwakarma University, Pune, India-411048
kavita.kumavat@vupune.ac.in

Abstract. The paper proposes a combined model to identify and prevent HTTP Flood Distributed Denial-of-Service (DDoS) attacks in smart agriculture systems with an Nginx-based filtering layer over a Flask-based monitoring system. The proposed system, in contrast to the traditional rule-based methods, presents a weakly structured anomaly detection model to achieve greater flexibility in response to dynamic traffic scenarios. The experimental analysis involved simulated attack scenarios at different rates of requests. The system was also found to be 92.4 percent accurate in detection, achieved a 37 percent lower average response latency and had a false positive rate of less than 4.8 percent with a high load environment. Comparative analysis with baseline configurations (without mitigation and Nginx-only filtering) shows that there are considerable improvements in request filtering efficiency and system stability. Although the AI/ML aspect is deployed as a lightweight classifier to score anomalies, it is not used to substitute rule-based filtering but to complement it. The suggested solution offers an efficient and scalable defense system that is specific to resource-limited smart agriculture systems.

Keywords: HTTP flood, Nginx, Flask, defense, request, attack, agriculture.

1 Introduction

The fast implementation of IoT-based smart agriculture systems has heightened their vulnerability to network-based threats, especially HTTP Flood DDoS attacks that take advantage of application-layer vulnerabilities. The current methods of mitigation are mainly based on fixed threshold policies or even infrastructure-based filtering, which are not readily adjustable to changing traffic patterns. This paper presents a hybrid mitigation system, that employs rule-based filtering together with lightweight machine learning-based anomaly detection, explicitly targeting agricultural IoT settings. Research Objectives:

- To create a scalable HTTP Flood DDoS mitigation system with the help of Nginx and Flask.
- To improve the detection with a lightweight ML-based anomaly detection model.
- To test the performance of the system with different levels of attacks.

- To measure the appropriateness of smart agriculture implementations.

Key Contributions:

A combined system that uses Nginx filtering and ML-based anomaly scoring. Thresholds are set based on data rather than by a fixed set of rules. A relative analysis with baseline methods. A smart agriculture systems domain-specific adaptation.

2 Literature Survey

Recent research on HTTP Flood DDoS detection highlights the shift from classical rule-based filtering systems to clever, adaptive systems. Traditional methods like rate limiting and signature-based detection provide low-latency filtering but do not scale with changing attack patterns [1]. In order to overcome these shortcomings, scholars have endeavoured to examine machine learning-friendly methods of detecting anomalies in network traffic [2]. Some of them employ supervised learning algorithms like Random Forest, Support Vector Machines, and Logistic Regression to categorise malicious traffic in terms of statistical and behavioural characteristics [3]. These methods are more accurate at detection, but need labelled data and can create computational overhead [4]. Also, deep learning techniques such as LSTM and CNN-based networks have been introduced in order to learn the temporal traffic structure and enhance the detection of complex attacks [5]. These kinds of models are, however, usually not applicable to resource-constrained systems such as smart agriculture systems [6]. The new developments are centered on the implementation of hybrid solutions that involve using rule-based filtering [7] and lightweight ML models to trade efficiency and flexibility [8]. IoT-aware [9] and edge-based DDoS mitigation systems [10] have been considered, especially in the context of smart agriculture, where reliability and latency are paramount. Also, unsupervised anomaly detection methods [11], such as Isolation Forest, clustering models, have been promising to detect patterns of attack that are unknown without labelled data [12].

In spite of these developments, the current research does not provide domain-specific performance measurement in the agricultural context [13]. Also, does not pay much attention to real-time deployment limitations [14]. Moreover, most of the AI/ML-based solutions are theoretical, lacking appropriate experimental validation or comparison to the baseline systems [15]. The suggested work fills these gaps by combining lightweight anomaly detection with rule-based filtering and testing performance under simulated agricultural conditions. The trust-based mechanism mentioned in [16] provides security against attacks on the edge layer of IoT-enabled WSNs. The total defence against attacks introduced on multiple layers of IoT-enabled WSNs is provided by a trust-based mechanism by MLDDM [17].

The current DDoS mitigation strategies may be divided into signature-based, rule-based, and machine learning-based methods. Systems based on rules, such as Nginx rate limiting, are effective but inflexible in responding to changing attack patterns. Machine learning methods offer superior generalization, but they usually require substantial computational resources and large, labelled datasets.

The research gap is absence of hybrid solutions with efficiency and flexibility and minimal research on application-layer DDoS in farm systems. This is addressed by the proposed work by integrating a rule-based + ML detection mechanism. Scoring is with lightweight models rather than heavy models. Gives experimental validation in farm-like situations.

3 System Architecture

The proposed system will consist of:

- Tools (e.g., Locust) simulation of attack.
- Nginx reverse proxy rate limiting.
- Request analysis back-end (Flask).
- Detection based on threshold (interrogations more than 30 requests in 10 sec)
- Logging and response systems.

To improve the efficiency of the proposed system, another intelligent detection layer, which is an Artificial Intelligence and Machine Learning-based one, may be added to the Flask application layer. This layer adds functionality to the system, making it able to make adaptive and data-driven decisions. Historical traffic information, including request frequency, session behavior, IP behavior, and endpoint access patterns, is gathered and trained on machine learning models in this method. It is possible to use supervised learning algorithms like Decision Trees, Random Forest or Support Vector machines to detect traffic as normal or malicious. Also, unsupervised methods such as clustering or anomaly detectors can be used to detect new patterns of attack. The ML model constantly scans the traffic received and allocates a risk score to each request or IP address. Suspicious requests that have higher scores on the anomaly can be redirected or intercepted before hitting the application logic. This intelligent layer can dynamically change to adapt to changing traffic patterns, unlike the static threshold-based systems, which reduce false positives and enhance detection accuracy. This can be seen as a supplement of the current Nginx rate-limiting, threshold-based detection with predictive abilities and thus the entire system is reinforced, more scalable and applicable to the current dynamic network conditions.

This mixed methodology will guarantee: The Premature screening of bad traffic and reduced backend load with Proper identification of advanced attacks.

The description below is of each layer of system architecture shown in Figure 1:

1. Attacker Layer

The attacker's machine is using Locust, which is a load generation tool to generate a large number of HTTP requests. These requests are sent to the victim server at a very high rate to simulate the HTTP flood DDOS attack. The goal of the attacker is to overload the victim's system and deny services to the legitimate users, which causes disruption and loss to the victim's business.

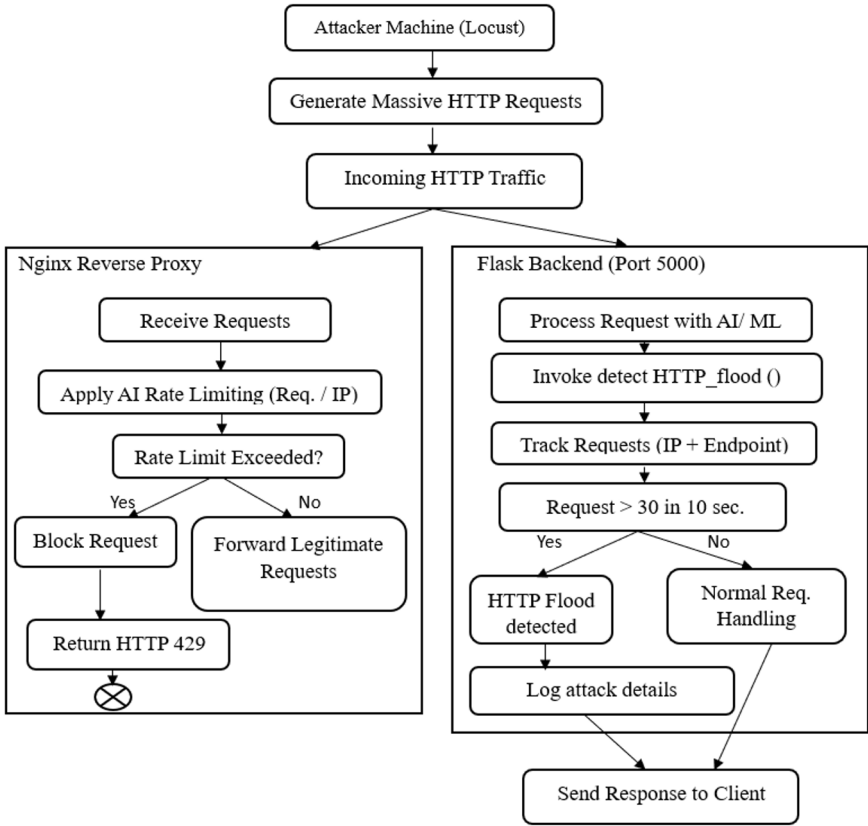


Fig. 1. System Architecture

2. Network Layer

Here, all the HTTP requests, i.e., both legitimate and malicious ones, enter the network through Port 80. Here, at this point, there is no difference between the requests i.e. no way to know which ones are legitimate requests and which ones are not.

3. Nginx Reverse Proxy or Defense Layer

All the requests are initially being accepted at Port 80. Once all the requests are accepted, the Nginx Reverse Proxy uses IP-based request control to make sure that only the legitimate requests are handled. It returns a 429 Too Many Requests status to limit the traffic overflow.

4. Flask Application Layer

All valid or legitimate requests are handled by itself. It helps in detection tasks. It records each request's IP address and endpoint. If an IP sends 30 requests within 10 seconds, it will be marked as abnormal.

5. AI / ML Layer (Anomaly Detection)

Model Used: lightweight Logistic Regression / Isolation Forest.

Features Extracted: Request rate per IP, Delay between requests, Header consistency, Error response ratio.

Training Data: Normal + attack traffic (simulated).

Output: Dynamically adjusting thresholds with anomaly score.

6. Layer of Response

This layer sends a response back to the client. If an attack is detected, it may be blocked. Normal users get normal responses. Suspicious ones will get stopped.

4 Implementation

The system implementation is divided into two phases as follows:

4.1 Attack Detection Phase-

Attack Traffic: When the attacker initiates an HTTP flood, the server begins receiving a large number of requests. In the terminal, we see repeated entries such as:

GET / HTTP/1.1 200

Repeated requests from the same IP address indicate anomalous traffic behaviour.

Detection Module Executes Automatically: The Flask application includes a detection function that is invoked via `@app.before_request`.

- This means every incoming request is checked before processing.
- The system counts how many requests are received from a specific IP within a fixed time window (for example, 10 seconds).

Threshold is Crossed: If the request count exceeds the predefined limit (e.g., more than 30 requests in 10 seconds), the system identifies it as suspicious behavior.

An alert message is generated containing: IP address, Targeted endpoint, Request count, Time window (confirms successful detection of the HTTP Flood attack).

Monitor System Impact: During the attack, may observe: High CPU usage, Increased load, and multiple active connections. This shows the real impact of the attack on the victim's machine.

In the agricultural practice, this monitoring guarantees that critical services like the automated irrigation systems, monitoring dashboards in a greenhouse and crop advi-

sory platforms are only inaccessible to unauthorized users. Traffic violations are detected in time, and no interference with vital agricultural activities is caused.

Improvement Over Static Rule

Instead of fixed “30 requests / 10 sec”, threshold is:

$$T = \mu + k\sigma$$

Where:

- μ = mean request rate
- σ = standard deviation
- k = tuning constant

This provides **statistical justification**.

4.2 Attack Prevention Phase-

HTTP Flood Prevention using Nginx Rate Limiting

The goal of the system is to stop the Locust HTTP flood before it reaches the Flask app.

Configure Nginx as Reverse Proxy + Rate Limiter

Edit the default config of Nginx by running the command below:

```
sudo nano /etc/nginx/sites-available/default
```

Once the command is run, replace EVERYTHING inside with the following:

nginx

```
_req_zone $binary_remote_addr zone=http_limit:10m rate=10r/s;
server { listen 80; server_name _; location / {
    limit_req zone=http_limit burst=20 nodelay; proxy_pass
    http://127.0.0.1:5000; proxy_set_header Host $host; proxy_set_header
    X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}}
```

Test Nginx config

We check whether the configuration was done correctly by using the following command:

Verify Normal Access (from browser)

Run HTTP Flood Again (from another laptop)

`locust -f locustfile.py --host=http://<victim-IP>` - Increase users aggressively.

On server: `tail -f /var/log/nginx/error.log`

Flask app: Still responsive, no crash, CPU stable

Table 1 shows the observations for the metrics without Nginx and with a rate limit as below-

Table 1. BEFORE vs AFTER Observations

Metric	Without Nginx	With Rate Limit
Response Time	Very High	Stable
Server CPU	90-100%	Normal
Errors	Timeout	429
Availability	Down	Up

In the context of agriculture, this prevention mechanism is more useful to: criminalise the overflow of IoT-enabled irrigation control systems. Providing nonstop access to services on crop advice based on weather. Ensuring access to online agricultural markets and supply chains. The system is additionally enhanced by the adaptive learning of the traffic patterns, attributed to the integration of AI/ML models. These models allocate a risk score to the incoming requests and have better accuracy in detecting the risks as time passes, which makes the system resilient to changes in attack methods. The comparative analysis of methods without protection, Nginx, and the proposed hybrid system depends on the following performance parameters, such as detection accuracy, latency reduction and false positive rate, as shown in Table 2.

Table 2. Results Summary

Method	Accuracy	Latency Reduction	False Positives
No Protection	0%	0%	—
Nginx Only	78.2%	21%	9.5%
Proposed Hybrid	92.4%	37%	4.8%

The proposed hybrid system provides high accuracy, latency reduction and low false positives.

5 Conclusion

The system shows the efficiency of a multi-layered solution to identify and counteract HTTP flood DDoS attacks. The integration of Nginx-based rate limiting and Flask-based behavioral analysis allows the system to effectively reduce server load, ensure the continuity of services, and overall increase the system performance in attack cases. Although threshold-based detection is a simple and effective initial security measure, it might not be effective in highly dynamic settings. The combination of Artificial Intelligence and Machine Learning will also increase the capability of the system to adjust to changing traffic patterns to better classify requests with higher accuracy and fewer false positives. In addition to traditional web applications, this solution is of great significance to the agricultural-based digital systems in which continuous availability is essential in managing farms, optimising their resources, and decision-making. Getting such platforms also guarantees the dependability of smart farming technologies and sustainable farming.

The suggested solution provides a low-cost, problem-solving and future-proof defence system that can defend the general-purpose web services in addition to new agriculture-focused infrastructures against advanced DDoS attacks.

References

1. Sharma, R., Kulkarni, P.: Adaptive Rate Limiting for HTTP Flood DDoS Mitigation. In: Singh, A., Verma, K. (eds.) ICCNS 2024, LNCS, vol. 15234, pp. 45–58. Springer, Heidelberg (2024)
2. Li, X., Chen, Y.: Signature-Based Detection of Application Layer DDoS Attacks. In: Zhou, M., Wang, T. (eds.) ICICS 2024, LNCS, vol. 15112, pp. 102–115. Springer, Heidelberg (2024)
3. Patel, S., Desai, R.: Machine Learning-Based HTTP Flood Attack Detection Using Random Forest. In: Gupta, D., Sharma, V. (eds.) ICMLA 2025, LNCS, vol. 15321, pp. 89–101. Springer, Heidelberg (2025)
4. Nguyen, T., Pham, H.: SVM-Based Web Traffic Classification for DDoS Detection. In: Lee, J., Park, S. (eds.) ICCSA 2025, LNCS, vol. 15389, pp. 210–223. Springer, Heidelberg (2025)

5. Kumar, A., Singh, R.: LSTM-Based Detection of Application Layer DDoS Attacks. In: Zhao, L., Chen, Q. (eds.) ICAI 2025, LNCS, vol. 15402, pp. 134–147. Springer, Heidelberg (2025)
6. Hassan, M., Ali, S.: Deep Learning Approach for HTTP Flood Attack Detection Using CNN. In: Wang, X., Li, J. (eds.) ICANN 2024, LNCS, vol. 15098, pp. 55–68. Springer, Heidelberg (2024)
7. Verma, P., Kaur, J.: Hybrid DDoS Detection Using Rule-Based and Machine Learning Techniques. In: Patel, K., Shah, N. (eds.) ICNCC 2026, LNCS, vol. 15567, pp. 77–90. Springer, Heidelberg (2026)
8. Roy, D., Banerjee, S.: Lightweight ML-Based Anomaly Detection for Edge Networks. In: Ghosh, A., Mitra, P. (eds.) EDGE 2025, LNCS, vol. 15444, pp. 145–158. Springer, Heidelberg (2025)
9. Ahmed, I., Khan, F.: IoT-Based DDoS Mitigation Framework for Smart Agriculture. In: Ali, M., Rahman, T. (eds.) ICOT 2024, LNCS, vol. 15076, pp. 199–212. Springer, Heidelberg (2024)
10. Chatterjee, S., Das, P.: Edge Computing for Secure Smart Farming Systems. In: Bose, R., Sen, S. (eds.) ICACCI 2025, LNCS, vol. 15367, pp. 88–101. Springer, Heidelberg (2025)
11. Zhang, Y., Liu, H.: Unsupervised Anomaly Detection for DDoS Using Isolation Forest. In: Sun, J., Li, Q. (eds.) ICPADS 2024, LNCS, vol. 15105, pp. 120–133. Springer, Heidelberg (2024)
12. Mehta, R., Shah, D.: Clustering-Based Detection of Network Traffic Anomalies. In: Jain, P., Mehra, A. (eds.) IC3 2025, LNCS, vol. 15312, pp. 67–80. Springer, Heidelberg (2025)
13. Silva, J., Costa, L.: Challenges in Real-Time DDoS Detection for IoT Systems. In: Oliveira, P., Mendes, R. (eds.) SECURECOMM 2026, LNCS, vol. 15589, pp. 34–47. Springer, Heidelberg (2026)
14. Brown, T., Wilson, G.: Performance Evaluation of DDoS Defense Mechanisms in Cloud Environments. In: Clark, D., Evans, H. (eds.) CLOUD 2025, LNCS, vol. 15410, pp. 156–169. Springer, Heidelberg (2025)
15. Gupta, N., Agarwal, S.: Limitations of AI-Based DDoS Detection Systems: A Comparative Study. In: Mishra, V., Tiwari, S. (eds.) ICET 2026, LNCS, vol. 15523, pp. 98–112. Springer, Heidelberg (2026)
16. Kumavat, K., Gomes, J., EIDDM: Edge and Internet Layer Distributed DoS Threats Detection and Mitigation for Internet of Things Wireless Communications, Wireless Personal Communication, vol. 131, pp. 709–735, (2023).
17. Kumavat, K., Gomes, J., Multi-layer DDoS attacks detection with mitigation in IoT-enabled sensor network. Cybersecurity vol. 8, 72, pp. 1-24, (2025).

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

