**DDoS Detection Using Entropy Computing**

**Abstract**

Distributed Denial of Service attacks are a potential threat to web infrastructure, whereas the service availability is targeted by malicious traffic overpouring of servers. Modern DDoS attacks are highly dynamic in nature and typical detection mechanisms can barely handle such cases, thus typically ending up with false negatives or positives. This paper introduces an entropy-based novel method for detecting DDoS attacks. By quantifying randomness in packet distributions, entropy is used as an effective measure to flag traffic anomalies. The technique has shown reliable detection even against highly skewed or unpredictable attack traffic.

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Topic** | **Page No** |
| 1 | Introduction | 3 |
| 2 | Existing Method | 3 |
| 3 | Proposed Method with Architecture | 4 |
| 4 | Methodology | 5 |
| 5 | Implementation | 5 |
| 6 | Conclusion | 6 |

**Introduction**

Distributed Denial of Service attacks are the most common form of cyber threats; they can cause extensive financial losses, service disruptions, and reputational damage. Distributed Denial of Service attacks function by flooding a target server with an overwhelming number of requests generated from a botnet of compromised systems, often running into thousands or millions.

**Why DDoS Detection Matters**

* Service Availability: Ensuring continuous availability of mission-critical services to authorized users.
* Business Continuity: Protects business organizations from financial and operational challenges.
* Reputation Management: Counteracting Public Trust Damage Caused by Service Outages.

Whereas various methods of detection exist, they usually fail in differentiating between legitimate high traffic and attack traffic. An entropy-based approach, on the other hand, is dynamic, lightweight, and real time, with packet distribution randomness analyzed.

**Existing Methods**

1. Threshold-Based Detection

Based on the fixed thresholds for metrics such as request rates or traffic volumes.

Limitations:

* Usually produces many false positives at times of legitimate traffic.
* Static thresholds can not adapt the dynamic change in the attacks.

1. Signature-Based Detection

Traffic patterns are compared with a database of known attack signatures.

Limitations:

* Totally ineffective for zero-day attacks or novel patterns.
* The signature database needs to be periodically updated.

1. Machine Learning Models

Supervised or unsupervised learning is used to classify traffic.

Limitations:

* High overhead in computation for real time detection.
* Requires extensive labeled datasets for training.

1. Statistical Analysis

Analyzes parameters such as inter-arrival times, packet sizes, or source distributions of traffic.

Limitations:

* Requires manual parameter tuning for different environments.
* It may not generalize across various traffic patterns.

**Proposed Method and Architecture**

**Proposed Method**

Entropy-based detection quantifies the randomness in packet distributions. Attack traffic, which is typically skewed, exhibits lower entropy values compared to normal traffic. This approach uses Shannon entropy as the primary metric:

H = -

where pi​ is the probability of each unique packet ID.

**Key Features**

1. Dynamic Baseline: Continuously calculates entropy for normal traffic to adapt to changes in legitimate behavior.
2. Anomaly Detection: Compares current entropy with the baseline to identify deviations.
3. Real-Time Processing: Provides immediate feedback on attack status.

**Architecture**

1. Traffic Simulation:

* Generates realistic normal and attack traffic distributions.

1. Entropy Computation:

* Computes baseline entropy for normal traffic.
* Calculates current entropy for incoming traffic.

1. Detection Module:

* Flags traffic as a DDoS attack if entropy deviation exceeds a pre-defined threshold.

**Methodology**

**Step 1: Traffic Simulation**

* Normal Traffic:
* Packet IDs are uniformly distributed between 1 and 10, reflecting legitimate randomness.
* Attack Traffic:
* Packet IDs are skewed, often restricted to a small subset (e.g., 1 and 2), to mimic botnet behavior.

**Step 2: Entropy Calculation**

* Baseline entropy is calculated using normal traffic.
* Current entropy is computed for the simulated or live traffic.

**Step 3: Deviation Analysis**

* The absolute deviation between baseline and current entropy is calculated.
* A threshold value (e.g., 0.5) is used to classify traffic.

**Step 4: DDoS Detection**

* Traffic is flagged as malicious if the deviation exceeds the threshold.

**Implementation**

**Tools and Technologies**

* Backend: Flask framework for API implementation.
* Frontend: HTML, CSS, JavaScript for user interaction.
* Libraries:
* math: For entropy computation.
* collections.Counter: For packet frequency analysis.
* flask-cors: To handle cross-origin resource sharing.

**Implementation Steps**

1. Simulate Traffic:

* Generate packet flows for both normal and attack scenarios.
* Use the function simulate\_packet\_flow() to create realistic distributions.

1. Compute Entropy:

* Implement Shannon entropy computation using calculate\_entropy().

1. Detect Anomalies:

* Compare current entropy with baseline entropy using detect\_ddos().

1. Integrate Frontend and Backend:

* Use Flask APIs to process traffic and deliver results.
* Build an interactive frontend for real-time detection.

**Conclusion**

Entropy-based detection provides a lightweight and effective solution for real-time DDoS attack identification. The system dynamically adapts to changes in legitimate traffic, reducing false positives while maintaining high detection accuracy. Future developments could involve:

* Testing the model with real-world datasets.
* Integrating the solution into a live Intrusion Detection System (IDS).
* Optimizing the system for large-scale deployment in cloud environments.

This approach not only addresses the limitations of existing methods but also offers a scalable and adaptive framework for combating evolving cyber threats.