DNS-Based Intrusion Detection System (DNS-IDS)

Celine John Philip (924165636), Kunjal Agrawal (924112058), Sakshi Singh (924160000)

1 Aim

The main goal of the current project is to build and implement a live DNS-IDS system that can understand the patterns of DNS traffic. Such a service will allow malicious activities like DNS tunneling, data theft, and anomalous behavior to be detected. The system uses a hybrid of statistical and rule-based methods to detect security loopholes in DNS queries while keeping the false alarm rate low. This initiative aims to provide a simple, handy, and resource-efficient way of using DNS traffic as a means of security.

2 System Overview

- **DNS Traffic Collector:** A Spring Boot app that listens to and saves DNS queries.
- **Feature Extraction:** Computes query rate, subdomain entropy, subdomain length, NXDOMAIN ratio, and many more.
- **Anomaly Detection Engine:** Identifies 5 different attack types using threshold-based rules and risk scoring.
- **Visualization and Alert Module:** React dashboard with the latest statistics and graphical representation of the threat.
- **System Testing:** Creation of a synthetic dataset with several attack patterns.

3 Motivation

DNS is fundamentally a major part of the internet system as it is today. However, attackers have come up with methods to conceal misleading data within these DNS queries, which has resulted in a series of situations where a large part of the internet has been shut down (2016 DDoS attack). The attack was an IoT botnet that used various insecure IoT devices to send DNS queries to the overloaded server of Dyn and thus caused the service of websites like Twitter and GitHub to be interrupted. These attacks were not only heavily talked about but were also heavily used because DNS was regarded as a very trusted component in most networks. Without resorting to a complicated or costly infrastructure, network security can be improved by utilizing DNS traffic as a means of detecting malicious activities. The reason behind our decision to delve into this topic is the review of the current attacks and the pressing need for the creation of a working solution that would overcome these challenges. The aim of building a DNS-based IDS is to be able to make the network less vulnerable to threats through simple, data-driven, and automated threat detection tools.

4 Literature Review

The Domain Name System (DNS) is the backbone of the Internet. Still, it is being increasingly leveraged by hackers for malicious activities, such as tunneling, data exfiltration, and command-and-control communication. Since DNS traffic is usually considered trustworthy and is allowed through firewalls, attackers can hide the encoded data in domain queries or responses in order to evade detection [1]. Hence, real-time monitoring of DNS traffic becomes a crucial defense tool in locating abnormal query behaviors and blocking covert communication channels.

Before the current situation, researchers have claimed that legitimate and malicious DNS traffic can be effectively separated by scrutinizing DNS query patterns and source content. Born and Gustafson

[2] asserted that natural language usage in legitimate domains is more easily predictable, while tunneling and algorithmically generated domains have a higher subdomain entropy, which indicates that the randomness of encoded data is reflected in these domains. In the same vein, studies have come up with signs that extremely long subdomains, high query rates, frequent NXDOMAIN responses, and unusual record types (e.g., TXT or NULL) are some of the clue words pointing at the presence of DNS tunneling and exfiltration attempts [3][4]. These statistical and behavioral features constitute the basis whereby an anomaly-based DNS intrusion detection system could be built.

More recent studies have exceeded the threshold set by statistics and have combined machine learning techniques to lower false positives and achieve detection accuracy. Hybrid methods that combine rule-based thresholds and model-driven approaches are reported to have improved performance in changing network environments, according to a comprehensive survey [5]. In the same way, a real-time detection framework proposed by MDPI Electronics [6] achieved very accurate results by taking entropy, query rate, and subdomain analysis as input features. Therefore, these findings, in their entirety, support our approach of using entropy, query rate, subdomain length, and NXDOMAIN ratio as the main features for the detection of anomalies in real-time DNS traffic..

5 Data and Methodology

The foundation of any intrusion detection system lies in the quality and comprehensiveness of the data. DNS-based threat detection requires the capture of detailed metadata about each DNS transaction to enable multi-faceted behavioral analysis. Our dataset was informed by industry best practices, particularly the comprehensive approach Datadog takes with regards to DNS log analysis for anomaly detection [7]. Indeed, their research demonstrates that DNS logs bear extensive metadata about each request and response which, if correctly analyzed, can show malicious patterns that may not be visible with traditional network security tools.

Each DNS query record in our dataset comprises the following pre-generated fields:

- **timestamp:** Unix timestamp indicating when the DNS request occurred.
- **client_ip:** IP address of the querying client. Helps identify malicious hosts, correlate traffic with threat intelligence, and track source-level attack behavior.
- **client_port:** Source port of the request.
- **query_name:** The requested domain name (FQDN).
- **query_type:** DNS record type (e.g., A, AAAA, TXT, ANY). Certain types can indicate malicious intent—for example, TXT records are commonly used in data exfiltration, and ANY queries are frequently leveraged in amplification attacks.
- **query_size:** Size of the query in bytes. Abnormally large queries may contain encoded data for exfiltration.
- **protocol:** Transport protocol (UDP/TCP).
- **response_code:** DNS server response code (0 = NOERROR, 3 = NXDOMAIN, etc.).
- **answer_count:** Number of resource records in response. Multiple answers with large sizes may indicate amplification targets.
- **raw_length:** Size of the DNS response in bytes. Values > 512 bytes indicate amplification potential; values > 1000 bytes are critical indicators.

5.1 Data Generation

We implemented our data generation framework as a Spring Boot RESTful API that programmatically generated realistic DNS query patterns for six distinct categories of traffic.

- **POST /api/dataset/generate?queryCount={n}** — Generates synthetic DNS datasets of configurable size.

Each time a client makes a DNS lookup request, it first goes to the DNS resolver. The resolver then interacts with other DNS servers recursively to get the right IP. Our goal was to figure out which attacks the type that could cause the DNS Resolver to be overwhelmed. So, we made fake traffic which looked like an overload-based attack. The dataset consists of the following categories:

1. **DNS Flooding Attacks:** Simulate a volumetric DDoS behavior pattern designed to exhaust the targeted DNS infrastructure by very high query rates.
2. **NXDOMAIN Flood Attacks:** Generate an attack scenario, in which the attacker sends a large number of DNS queries for non-existent domain names thus the resolver is forced to consume its memory unnecessarily.
3. **Random Subdomain Attacks:** Create a huge number of different, non-existent subdomains so as to get round the cache and thus the DNS resolvers and authoritative servers become overloaded because of repeated upstream lookups.
4. **DNS Amplification Attacks:** Simulate perpetration of DDoS attacks typical of reflection scenarios by having small request sizes (40–60 bytes) with correspondingly large response payloads (600–1500 bytes) that result in high amplification ratios.
5. **DNS Data Exfiltration (Tunneling) Attacks:** Represents the situations in which attackers send small DNS queries that trigger very large responses from open resolvers. For this purpose, the attackers use extremely long subdomains, excessive usage of the TXT record, and high entropy values (> 4.5).
6. **Normal Baseline Traffic:** Describes the legitimate DNS activities performed by a typical user and the applications under usual circumstances. This category makes behavioral ground truth which enables the system to identify benign traffic from the malicious one and also reduces the number of false positives during the evaluation.

5.2 System Architecture

DNS-IDS was implemented as a multi-tier Spring Boot application following a modular service-oriented architecture to ensure scalability, maintainability, and extensibility. The system comprises four primary layers:

- **Data Layer:** Utilizes H2 in-memory database with JPA/Hibernate ORM for high-speed query storage and retrieval. The `DnsQueryEntity` model encapsulates all DNS query metadata with appropriate indexing on `client_ip` and `timestamp` fields to optimize aggregation queries during analysis.
- **Service Layer:** Provides the essential detection logic via two main services:
  - `DataGenerationServiceImpl:` Handles synthetic dataset creation with configurable query counts and proportional attack distribution.
  - `AnalysisServiceImpl:` Runs multi-pattern threat detection algorithms, including feature extraction and risk assessment.
- **Controller Layer:** Exposes RESTful API endpoints for frontend integration.
- **Presentation Layer:** Integrated web dashboard providing real-time visualization of detected threats, risk scores, and mitigation recommendations.

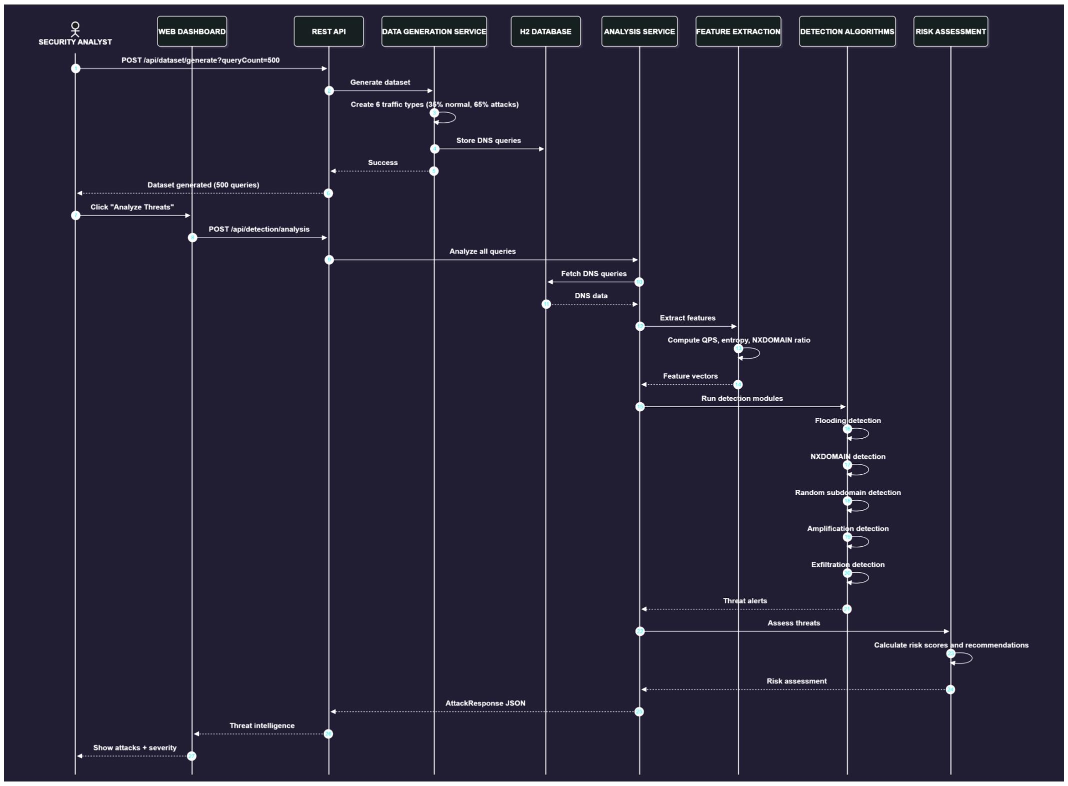


Figure 1: Multi-tier architecture of the DNS-IDS system.

6 Methodology

6.1 Feature Extraction Engine

To transform raw DNS query metadata into actionable security intelligence, we implemented a comprehensive feature extraction pipeline computing several distinct behavioral metrics.

6.1.1 Temporal Features

1. **Query Per Second Calculation (QPS):** For each unique source IP address, we calculate the query rate by first determining the time window and then calculating the queries per second.

$$\text{timeWindow} = \max(\text{timestamp}) - \min(\text{timestamp}) + 1$$
$$\text{queriesPerSec} = \frac{\text{queryCount}}{\text{timeWindow}}$$

2. **NXDOMAIN Ratio Computation:** We calculated NXDOMAIN Ratio by counting all queries with `responseCode = 3` for each IP and then computing:

$$\text{nxdomainRatio} = \frac{\text{nxdomainCount}}{\text{totalQueries}}, \quad \text{nxdomainPerSec} = \frac{\text{nxdomainCount}}{\text{timeWindow}}$$

3. **Base Domain & Subdomain Extraction:** We extract the base domain by taking the last two labels of the FQDN and derive the subdomain by removing this base domain. This allows counting unique subdomains and identifying high-uniqueness patterns indicative of random subdomain attacks.

4. **Shannon Entropy Calculation:** We build a frequency map of all characters in the subdomain, calculate the probability for each unique character, and apply the Shannon formula to compute

entropy. Subdomains with entropy > 4.5 are considered potentially malicious, indicating DNS tunneling or data exfiltration:

$$H(X) = -\sum p(x_i) \cdot \log_2 p(x_i)$$

5. **Subdomain Length Analysis:** We measure the character count of extracted subdomains. Legitimate subdomains typically contain 5–30 characters, while data exfiltration attempts encode information in longer subdomains.

6. **TCP Fallback Ratio:** Sometimes, a spike in TCP traffic indicates that DNS responses are too big to fit within UDP. In such cases, the resolver falls back on TCP to retrieve the complete answer. We calculate the TCP fallback ratio as:

$$\text{tcpRatio} = \frac{\text{tcpCount}}{\text{totalQueries}}$$

7. **Amplification Ratio Calculation:** We compute the response-to-query size ratio to identify amplification attack potential.

8. **ANY Query Detection:** We count and calculate the ratio of ANY query types, historically abused for maximum amplification. ANY queries request all available record types, yielding large responses—a key indicator when combined with high amplification ratios.

9. **Base64/Base32 Pattern :** We used looksLikeBase64() to determine whether subdomains matched encoding patterns, identifying strings with >90% alphanumeric characters and Base64-compatible character sets.

6.2 Multi-Pattern Detection Algorithm

DNS IDS is able to detect a wide range of DNS-based attacks by using a multi-pattern detection engine that sequentially invokes a set of analysis modules, each of which is specialized in a different threat pattern. Each module scrutinizes temporal behaviors, statistical anomalies, and semantic features derived from pre-processing in an attempt to produce threat alerts if the thresholds are overstepped. The multi-pattern detection engine has a look at every DNS log batch from five behavioral dimensions:

- **Query Rate (QPS):** We calculated `QPS` [6.1.1] and if it exceeded the flooding threshold of 100 queries/second, we flagged the IP as potential attacker. Based on the QPS rate, we calculated the risk factor and severity. Higher QPS indicated more aggressive attacks requiring immediate attention.
- **NXDOMAIN Anomalies:** We computed `nxdomainRatio`, `nxdomainCount` and `nxdomainPerSec` [6.1.1]. If `nxdomainPerSec` ≥ 20 queries/sec or `nxdomainRatio` $\geq 70\%$ and `nxdomainCount` > 10, it indicated NXDOMAIN Attacks.
- **Subdomain Randomness:** We counted the number of distinct subdomains queried by a single IP address for a specific base domain(uniqueCount) and calculated the percentage of queries that are unique(uniquenessRatio). If `uniqueCount` ≥ 30 and `uniquenessRatio` $\geq 80\%$, indicated Random Subdomain Attack.
- **Amplification Behavior:** We detected amplification if `largeResponseRatio`$\geq 50\%$, `anyQueryCount`$\geq$ 10% , high TCP fallback ratio and high Amplification ratio.
- **DNS Data Exfiltration Indicators:** Four detection patterns were put into practice: (1) long subdomains longer than 50 characters that indicate encoded payloads; (2) Shannon entropy analysis, where entropy > 4.5 indicates randomness typical of encrypted/encoded data; (3) TXT query abuse

(`txtQueryCount` $\geq$ 15 or `txtQueryRatio` $\geq$ 40%) because TXT records carry arbitrary payloads; and (4) Base64/Base32 pattern detection that identifies encoded strings. Additionally, we looked for data fragmentation by checking whether `uniqueSubdomains` $\geq$ 40 per base domain. The IP was marked for DNS data tunneling/exfiltration if any pattern was found.

For simple understanding, we have a diagram particularly for the detection algorithm.

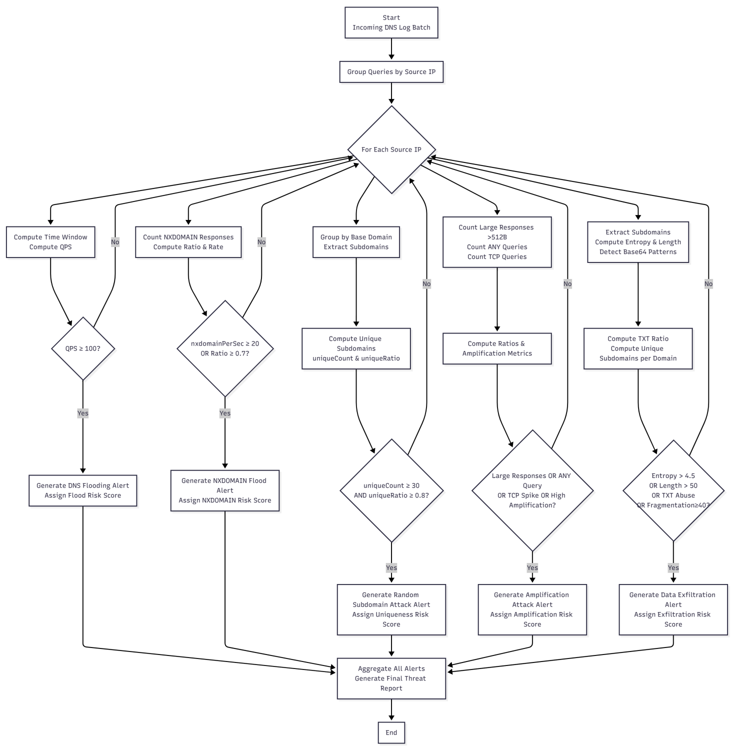


Figure 2: Detection Algorithm Diagram

Each module feeds its score and alert into a comprehensive, multilayer DNS threat-detection architecture, enabling detection of diverse attack types in a coordinated and scalable manner.

6.3 Risk Assessment and Threat Classification

We developed a thorough risk assessment framework that unifies various threat indicators into cohesive, actionable threat scores so that security teams could efficiently prioritize incident response.

7 Results

We evaluated the DNS-IDS system using a synthetic dataset of 15,000 DNS queries generated across six traffic categories. The system successfully detected all five forms of DNS-based attacks present in the dataset, producing a total of 14 threat alerts, of which 9 were classified as critical. The analysis completed in approximately 1,261 ms (1.26 seconds), achieving a processing throughput of nearly 11,896 queries per second, confirming that the IDS can operate efficiently on high-volume DNS traffic. Among all attack types, Random Subdomain Attacks were detected most frequently, as this category generated the largest number of unique malicious queries and consistently triggered the uniqueness-ratio–based anomaly detector.

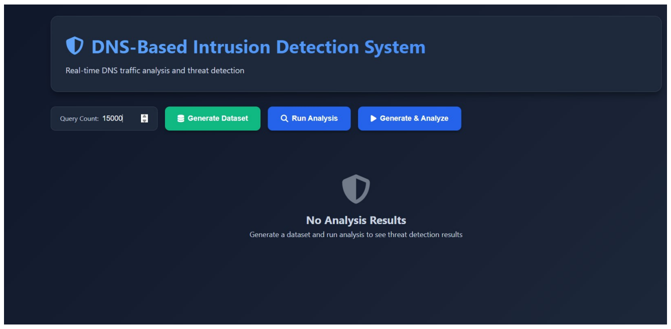
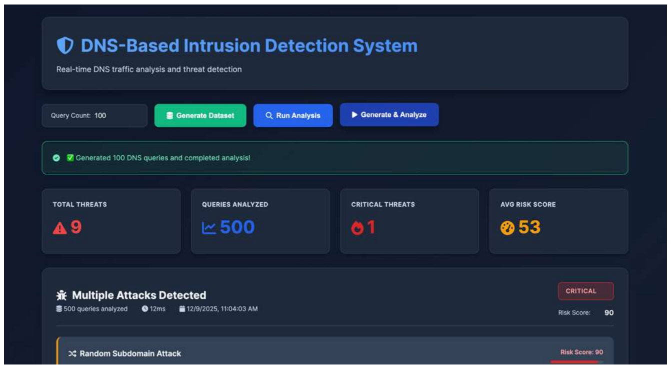

DNS-Based Intrusion Detection System
Real-time DNS traffic analysis and threat detection
Query Count: 15000
Generate Dataset
Run Analysis
Generate & Analyze
No Analysis Results
Generate a dataset and run analysis to see threat detection results

Figure 3: Home page



✅ Generated 100 DNS queries and completed analysis!

TOTAL THREATS	QUERIES ANALYZED	CRITICAL THREATS	AVG RISK SCORE
9	500	1	53

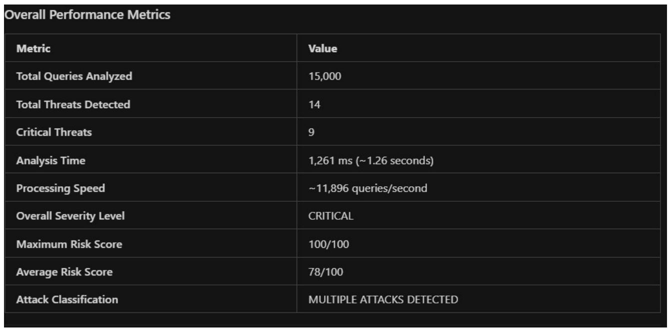
Multiple Attacks Detected

500 queries analyzed 12ms 12/9/2025, 11:04:03 AM

CRITICAL Risk Score: 90

Random Subdomain Attack Risk Score: 90

Figure 4: Detection page



Overall Performance Metrics

Metric	Value
Total Queries Analyzed	15,000
Total Threats Detected	14
Critical Threats	9
Analysis Time	1,261 ms (~1.26 seconds)
Processing Speed	~11,896 queries/second
Overall Severity Level	CRITICAL
Maximum Risk Score	100/100
Average Risk Score	78/100
Attack Classification	MULTIPLE ATTACKS DETECTED

Figure 5: Analysis Result

8 Code & Data Availability Statement

The code, along with all required dependencies, is provided here:

- `https://github.com/kunjal2002/ecs_235`

9 Use of Large Language Models (LLMs)

Our team developed the DNS-IDS system by manually researching, coding, and analyzing it while ensuring that both the methodology and the basic implementation were done directly. However, to improve output and gain new insights, we used LLMs in certain areas. LLMs helped with minor code optimizations and syntax issues, but we handled feature extraction, DNS log processing, and detection manually, as those are significant parts of the project. Importantly, every code optimization implemented by LLMs after integration was first checked and verified for accuracy and reliability. In addition to programming, LLMs provided minimal support in writing and proofreading. They helped style LaTeX text and improve technical descriptions for clarity. We used structured prompt engineering methods to keep control over AI-generated outputs. This included iterative refinement and focused queries to ensure relevance. To ensure correctness, we also manually checked all AI-assisted recommendations. We implemented strict human supervision because we were aware of the potential drawbacks of LLMs, such as hallucination and lack of technical details.

10 Future Work

The DNS-IDS system's accuracy, scalability, and practicality can all be improved in the future. In order to minimize false positives and automatically adjust to shifting traffic patterns, future iterations of the current design, which employs rule-based thresholds for detection, might include machine learning models like Random Forest or Isolation Forest. Instead of batch-based analysis, continuous monitoring at scale would be possible with a real-time streaming pipeline that makes use of tools like Kafka or Redis Streams.

11 Glossary

- **FQDN (Fully Qualified Domain Name):** A Fully Qualified Domain Name is the complete, unique address for a host or computer on the internet, specifying its exact location.
- **Entropy (Shannon Entropy):** A measure of randomness in a domain or subdomain string. Normal human-generated domains have entropy between 2.5–4.0, whereas values above 4.5 indicate highly random, encoded, or algorithmically generated domains, aligning with DGA detection research.

References

[1] GIAC, "Detecting DNS Tunneling," SANS Institute Whitepaper, 2012. [Link].

[2] K. Born and D. Gustafson, "Detecting DNS Tunnels Using Character Frequency Analysis," *arXiv preprint arXiv:1004.4358*, 2010. [Link].

[3] D. Tatang, F. Quinkert, N. Dolecki, and T. Holz, "A Study of Newly Observed Hostnames and DNS Tunneling in the Wild," ResearchGate, 2019. [Link].

[4] D. Nikou, "Detection, Analysis and Measurement of DNS Tunnelling Techniques," Master's Thesis, Radboud University and SIDN Labs, 2024. [Link].

[5] Y. Wang, A. Zhou, S. Liao, and R. Zheng, "A Comprehensive Survey on DNS Tunnel Detection," *Computer Networks*, vol. 197, p. 108322, July 2021. DOI: 10.1016/j.comnet.2021.108322. [Link].

[6] "Real-Time Detection System for Data Exfiltration over DNS," *Electronics*, vol. 12, no. 6, p. 1467, MDPI, 2023. [Link].

[7] "Monitor DNS log for Network and Security Analysis." [Link].

[8] "What is DNS Server." [Link].

[9] "How DNS Works?" [Link].

[10] "Introducing anomaly detection in Datadog?" [Link].