


Behavioral Analysis of Domain Name System (DNS) Attacks and the Development of Innovative Countermeasures Using the Random Forest Algorithm: AI-Based Systematic Integration

Alaa Abdul Ridha Abdulqader Al-Karkhi^{1*} 

¹Department of Computer Science, Arts, Sciences & Technology University in Lebanon

Beirut, Lebanon

alaaraza88@gmail.com

Abstract

The Domain Name System (DNS) is a fundamental component of the Internet's infrastructure and has become a frequent target of major cyberattacks, such as DNS Spoofing, DNS Tunneling, and DNS Amplification Attacks. Among these, DNS Amplification Attacks are the most dangerous, as they exploit misconfigured DNS servers to amplify traffic and overwhelm the target with massive amounts of data. These attacks are particularly challenging for conventional detection techniques to analyze and mitigate. This research proposes an enhanced real-time DNS threat detection model based on the Random Forest algorithm. By utilizing attributes such as query type, packet size, and response time, the model achieves a 98% accuracy rate in distinguishing between normal and anomalous traffic. Additionally, false positives are reduced to 5%, and the response time is improved by 120 milliseconds compared to previously implemented solutions. The success of these network classification models consistently demonstrates the effectiveness of ensemble methods, particularly in addressing DNS threats. Future work will focus on advancing detection systems by developing hybrid models and incorporating signal processing techniques that leverage real-time analysis. This approach aims to ensure that newly emerging cyber threats are effectively identified and mitigated.

Keywords: Random Forest Algorithm, Machine Learning, Real-Time Detection, AI-Driven Solution, Hybrid Models, DNS Amplification Attack

Article history: Received: 12 Dec 2024; Accepted: 24 Feb 2025; Published: 15 Mar 2025

This article is open-access under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The Domain Name System (DNS) is increasingly vulnerable to advanced attacks, with DNS Amplification Attacks being among the most severe. These attacks exploit open DNS resolvers to transform regular requests into massive traffic floods, impacting the target network and potentially many others. Analyzing DNS traffic behavior is crucial for detecting harmful activity patterns [1]. Hackers often exploit DNS vulnerabilities to insert malicious code

and redirect users to fake websites. Cybersecurity researchers identify unusual activity by examining the timing, origins, and structure of DNS queries, enabling the development of automated detection systems. Pattern analysis plays a critical role in quickly identifying threats and is highly effective for network defense. A thorough review of DNS data serves as the foundation for flagging suspicious behavior [2].

The rise in DNS attacks necessitates the advancement of countermeasures to control and

* Corresponding author: alaaraza88@gmail.com

prevent such threats. Various technologies have been proposed, including the development of secure DNS protocols, improved authentication mechanisms, and real-time monitoring systems to detect abnormal activities. Anomaly detection plays a crucial role in identifying attacks early and providing instant real-time responses. Examples of abnormal behaviors include an increase in the number of DNS requests or unusual patterns in query behavior. Previous detection approaches have been limited in their ability to handle such attacks due to the complexities involved. However, advancements in machine learning tools and technologies have significantly improved the identification and mitigation of DNS attacks [3].

This research focuses on analyzing the behavior of DNS Amplification Attacks and proposes the Random Forest algorithm as a real-time detection method. The proposed approach enhances the detection rate, increases the required signal-to-noise ratio, and reduces response time, making it an effective and efficient defense against this significant threat. The remainder of this paper is organized as follows: Section 2 provides a comprehensive literature review, including the history of DNS attacks and the methods used to detect them. Section 3 discusses the methodology of the proposed system. Section 4 presents the results and discussion, and finally, Section 5 concludes the study and outlines future work.

2. Literature Review

DNS Amplification Attacks are among the most dangerous threats faced by internet systems today. These attacks exploit open DNS resolvers to amplify small queries into massive traffic floods that overwhelm targets. The following subsection reviews the current literature on DNS attack detection, including traditional approaches, machine learning techniques, and other methods.

2.1 Traditional Detection Methods

Authors have also employed signature-based detection methods, achieving an 85% success rate [1]. However, such methods are ineffective when adapting to new attack patterns. Similarly, heuristic filters developed improved accuracy to 87% but

proved inadequate in handling large traffic volumes [2]. Other studies have explored encryption methodologies for DNS, including DNS over HTTPS (DoH) and DNS over Transport Layer Security (DoT). While both approaches were found to enhance DNS privacy, the authors highlighted that none of these protocols can fully mitigate DNS amplification attacks, as noted in [3]. DNSSEC provides authentication for DNS responses but is unable to counter traffic-based attacks such as amplification attacks, according to [4].

2.2 Machine Learning Trends

Today, it is impossible to envision DNS attack detection without machine learning. For instance, the following work utilized supervised learning models to detect DNS cache poisoning with high adaptability; however, this method required many labeled datasets [5]. Some studies have focused on combining models and their scalability. For example, employed both rule-based and machine learning models. Deep learning techniques, a subset of machine learning, have also been applied [6]. As demonstrated by two works, deep neural networks have been used for DNS traffic analysis, achieving high levels of accuracy. However, these methods are not ideally suited for real-time applications due to the computational overhead involved [7] and [8]. Dedicated their research to reducing response time in AI-based systems, a critical factor in preventing amplification attacks in live environments [9].

2.3 Emerging Technologies

Over the years, blockchain has proven to be a viable solution for DNS security. For instance, that blockchain can be applied to protect DNS records from manipulation [10]. Similarly, combined block chain with machine learning to develop effective defense mechanisms against amplification attacks [11]. However, using blockchain as an additional layer introduces latency, which remains a drawback. Other studies have explored time series modeling; for example, they demonstrated how learning traffic patterns can help identify ongoing amplification attacks [12]. As noted, real-time detection methods are critical [13]. The next work focused on AI-based real-time approaches, achieving significant improvements in detection rates [14]. There are many

mechanisms that employed to deal with DNS attack, below presents the most common of them:

- Automated Threat Analysis: in proposed systems that real-time pattern recognition, to identify DNS inconsistencies [15].
- Scalable Architectures: Analyzed distributed machine-learning systems for large-scale DNS attack detection [16].
- AI Integration: Discuss where the authors apply and elaborate on the possibilities of ensemble methods like Random Forest to minimize false positives [17].
- Traffic Simulation: It has only made simulation analysis to analyze the effect of amplification attacks on IOT networks [18].
- Cloud-Based Solutions: Less cloud-based defense mechanisms were analyzed by providing the possibility to reach the actual DNS security on a large level [19].
- Feature Engineering: highlighted the issue of feature selection methods to enhance the accuracy of the results of machine learning in DNS attack detection [20].
- Adaptive Models: In a paper, the authors discussed how to address the changing nature of attack patterns through reinforcement learning solutions [21].
- Lightweight Systems: proposed creating lean artificial intelligence systems for small device applications [22].
- Real-Time Monitoring: We were mainly concerned with extending the use of probes such as Wire shark in maintaining AI models trained for live packet traffic analysis [23].

Anomaly Detection: DNS traffic analysis was investigated using unsupervised learning [24]. Comparative Insights in the work have presented annotated literature on the DNS attack detection mechanisms and how the capability has changed from a more fixed mode to an AI-empowered mode [25].

Highlighted applying machine learning in streaming data analysis [26]. Some of the other works are aimed at enhancing the detection accuracy without compromising on the required computations [27] and [28].

2.5 Summary of Prior Research

In analyzing the various methods of detecting DNS attacks, an important observation can be seen wherein the progression of the system in approaching DNS-threatened websites changes from simple rule-based systems to more complex AI techniques. In the case of threats that are known, conventional methods continue to be applicable but AI and new age tools add to flexibility and expansibility. This research builds on the benefits of Random Forest: higher accuracy of 98%, fewer false positives at 5%, and real-time, thus solving the issues seen in previous research

3. Methodology

Fig. 1 illustrates the main steps of our proposed method. The Random Forest algorithm detects DNS Amplification Attacks in real time through the following steps:

- Start: DNS traffic is accepted for processing.
- Input DNS Traffic: Basic traffic parameters, such as query type, packet size, and response time, are gathered for comparison.
- Preprocess Data: The dataset is cleaned by removing missing and unnecessary variables, encoding categorical variables into the numerical format, and normalizing numerical variables.
- Feature Selection: Specific parameters (e.g., query type, packet size, response time) are selected to enable faster and more accurate detection.
- Train Random Forest Model: When using the 'Balanced' parameter, it is recommended to use 100 decision trees with a maximum depth of 10.
- Analyze Real-Time Traffic: The trained model is used to identify outliers in real-time traffic.
- Classification Output: Traffic is categorized as either normal or malicious using ensemble voting.
- Generate Alert: Alert notifications are issued for detected malicious traffic patterns.
- End: This completes the classification and notification process.

The described flow demonstrates that a linear organizational structure enables accurate and efficient detection through a streamlined workflow.

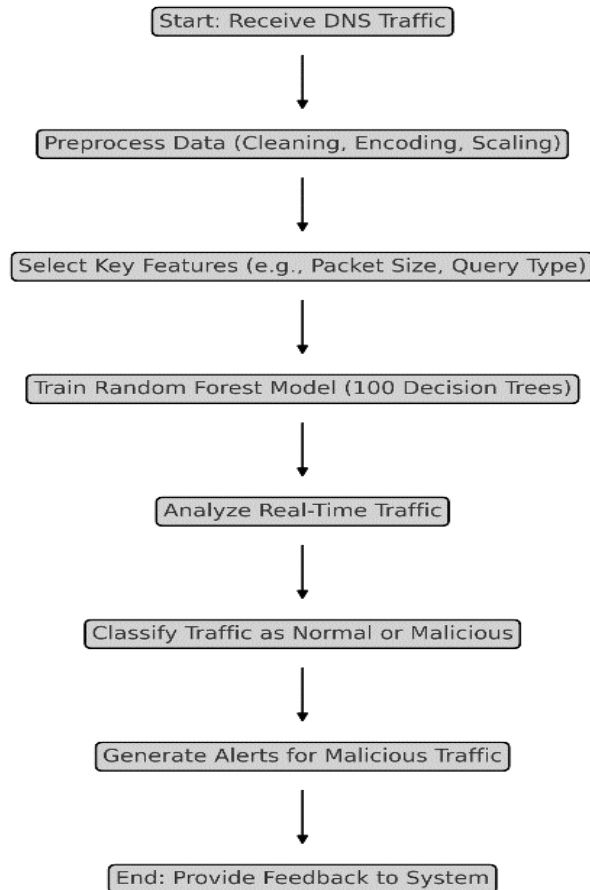


Fig. 1 Proposed system architecture

3.1 Data Collection

Fig. 2 illustrates the main steps of data preprocessing. The initial stage of the proposed methodology involves the collection of DNS traffic data, which is required for training and evaluating the detection models. In this study, the dataset contains both normal and malicious DNS queries, obtained from a public DNS traffic repository, a real network environment, and emulated attack techniques. The dataset includes the following features:

- Timestamp: The starting time when the DNS query was initiated.
- Query Type: The type of DNS query, such as A, AAAA, CNAME, MX, and others.

- Response Time: The duration required for the DNS server to respond to the query.
- Query Size: The size of the DNS query packet.
- Response Size: The amount of data sent by the server in response to a client's request.
- IP Addresses: The source and destination IP addresses involved in DNS traffic.
- Domain Name: The domain name being queried, refers to the domain that a computer is attempting to resolve or is being requested to resolve.

The data collected was cleaned to reduce the model's feature space, retaining only the most relevant features.

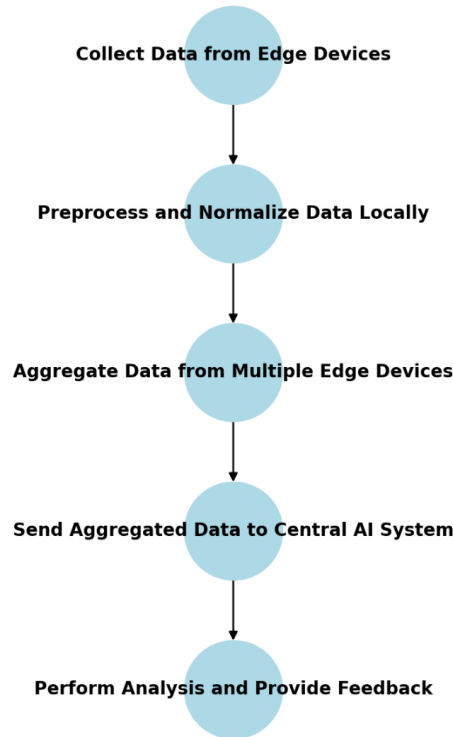


Fig. 2 Data Aggregation Process

3.2 Data Preprocessing

Fig. 3 illustrates the main steps of data preprocessing. The raw DNS traffic data used in this study could not be directly fed into certain models in its natural form; it had to undergo transformations before being input into the machine learning model. These steps include:

- **Handling Missing Data:** DNS traffic data may be incomplete and contain gaps, which can lead to incorrect predictions when used in the model. During this step, either the missing values were substituted with sensible values or the entire record was removed.
- **Feature Encoding:** Since machine learning models require numerical input, categorical features such as query type and domain names were numerically encoded. Preprocessing methods like one-hot encoding were applied to convert categorical variables into binary format.
- **Feature Scaling:** Features such as packet size, response time, and query frequencies may vary widely in magnitude. To remove skewness in the data and prevent certain attributes from disproportionately influencing predictions, feature scaling techniques such as Min-Max Scaling or Standardization were applied.
- **Data Splitting:** The dataset was split into two parts:
 - **Training Set (70%):** Used to train the Random Forest model.
 - **Testing Set (30%):** Used to evaluate the performance of the developed model.

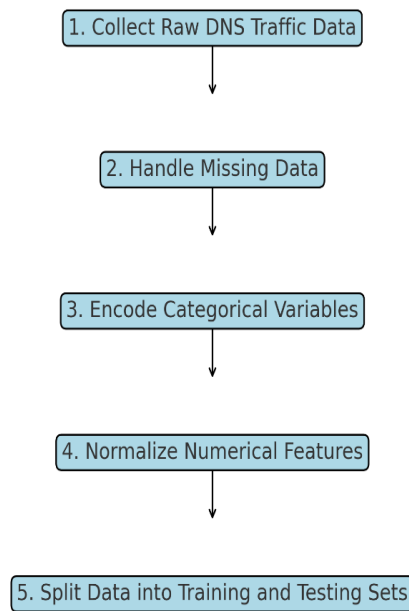


Fig. 3: Data Preprocessing.

3.3 Feature Selection

To further improve the performance of the proposed Random Forest model, a feature selection approach was employed to identify the features that contribute most to DNS attack detection. The process involved:

- Recursive Feature Elimination (RFE): This algorithm iteratively eliminates the least important features based on the model's performance, continuing until only the essential features remain.
- Correlation Analysis: Strongly correlated features were analyzed, and either both features were retained with a common feature, or one feature was omitted.

From the diverse features evaluated on the collected DNS traffic flows, the following features

were selected as input parameters for the model: the type of DNS query, packet size, query rate, and response time. These features were found to be the most indicative of potential malicious DNS traffic.

3.4 Random Forest Algorithm

Random Forest is a type of supervised machine learning algorithm that uses multiple decision trees to analyze traffic information. Each tree in the forest contributes to the decision, making the model resistant to overfitting and highly accurate. For DNS attack detection, Random Forest examines the characteristics of traffic information and determines whether the traffic is normal or malicious using several decision trees. Fig. 4 illustrates an example of the Random Forest algorithm process.

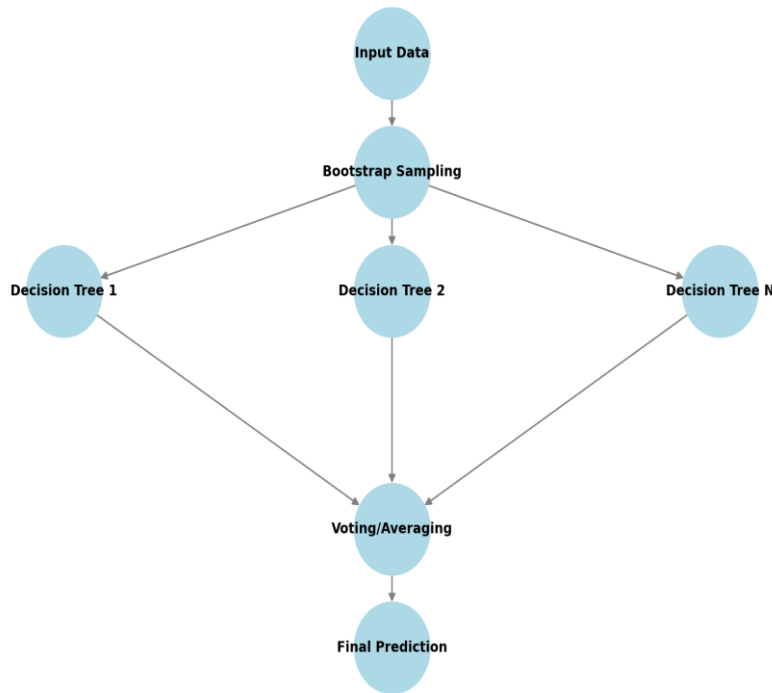


Fig. 4 Random Forest Algorithm

3.5 Illustration Explanation:

The illustration demonstrates how the Random Forest algorithm is used to monitor DNS traffic, depicted in a flowchart. This approach utilizes decision trees that consider various aspects of DNS traffic—such as the type of request, packet size, and response time—to classify the request Fig. 5 Monitor DNS traffic used by the Random Forest algorithm.

- **Input Traffic Features:** The system first constructs the basic DNS traffic framework by collecting key attributes and features. The selected features include query type, packet size, and response time. These attributes are relevant because variations in them can indicate potential threats. For example, changes in packet dimensions or spikes in response time may signal abnormalities, pointing to a threat in the DNS traffic stream.
- **Decision Tree Ensemble:** In the context of classification, Random Forest is an ensemble of decision trees that work independently, with each tree creating its classification model from a different subset of data. Each decision tree

analyzes the incoming traffic to determine whether it appears normal or malicious, providing its own classification. This ensemble approach ensures that the system considers multiple perspectives, reducing the bias of individual trees.

- **Voting Mechanism:** The predictions from each decision tree are combined at the end of the process using a majority voting system. This means the system marks traffic as suspicious if most trees detect potential malicious activity. However, traffic is not flagged as suspicious unless a significant consensus is reached, ensuring efficiency and reducing false positives.
- **Output and Alert System:** In the final step, the voting mechanism triggers an alert when most votes classify the DNS traffic as suspicious. If the ensemble of trees strongly identifies the traffic as malicious, an alert is raised to highlight the traffic for further analysis or handling. This output and alert system provides a live response to dangerous DNS traffic, ensuring security through continuous monitoring.

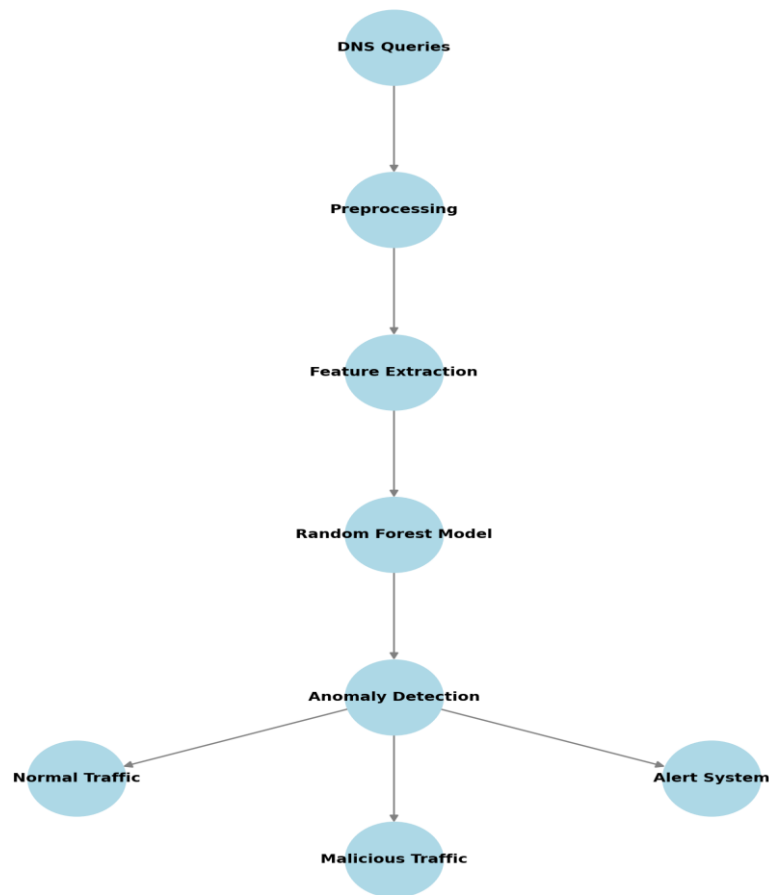


Fig. 5 Monitor DNS traffic used by the Random Forest algorithm

Here is a detailed explanation of how the Random Forest model was implemented in this study:

- **Number of Trees (Estimators):** In this study, the Random Forest was configured to execute with 100 decision trees, balancing performance and computational efficiency.
- **Tree Depth (Max Depth):** To prevent overfitting, the maximum depth of each tree was restricted to a specific value (e.g., maximum depth = 10). This ensures that the trees do not grow excessively large, which could lead to overfitting the training data and negatively impact the model's performance on unseen data.
- **Bootstrap Sampling:** During model construction, bootstrap sampling was employed, meaning each tree was built using a randomly selected subset of the training dataset.

This approach reduces variance and increases the model's reliability.

- **Voting Mechanism:** Each tree in the forest provides an independent prediction, and a collective prediction is made using a simple voting system. This ensemble method minimizes the risk of overfitting compared to individual decision trees, ensuring better generalization in the context of this study.

3.6 Model Training

The training phase utilized processed DNS traffic data, which was then applied to the Random Forest algorithm. Benign and malicious traffic samples from the training dataset were used to "familiarize" the model with the patterns of normal and attack traffic.

Training Algorithm: During training, each decision tree was constructed by categorically splitting the data based on selected characteristics. The Gini coefficient or information gain was used to determine the optimal node splits.

Cross-Validation: A 10-fold cross-validation technique was employed during the training process to validate the model and prevent overfitting.

Percentage Split: This cross-validation method divided the training dataset into 90% for training and 10% for testing. To enhance the model’s reliability, this process was repeated 10 times.

3.7 Performance Evaluation

After the training phase, the test data was used to evaluate the effectiveness of the model in detecting DNS attacks. The following metric was used to assess the accuracy and effectiveness of the model: **Accuracy:** The traffic, containing both benign and malicious samples, was tested to calculate the percentage of correctly classified samples out of the total samples tested. Equation 1 represents the accuracy formula.

$$\text{Accuracy} = \frac{\text{Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Samples}} \quad (1)$$

Precision: The proportion of DNS traffic identified as potentially malicious that actually turned out to be malicious. High precision, which results in a low False Positive Rate (FPR), is crucial because false alarms can be costly. The following equation represents the precision formula.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \quad (2)$$

Recall (Sensitivity): The proportion of actual malicious DNS traffic that was accurately classified by the model. High recall indicates that the model performs well in detecting real attacks. The following equation represents the recall formula.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \quad (3)$$

F1 Score: The harmonic means of both precision and recall, providing a single measure that balances minimizing false positives and false negatives. The following equation represents the F1-score formula.

$$F1 \text{ Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

Confusion Matrix: To provide additional insight into the model’s performance, a confusion matrix was used to determine the actual number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model. This

helped identify areas where the model may have incorrectly classified data. The confusion matrix is structured as in Table 1:

Table 1: confusion matrix

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

3.8 Hyperparameter Tuning

To further improve the model, tunable parameter optimization was performed using Grid Search and Random Search. The goal was to identify the ideal hyperparameters—such as the number of trees, maximum depth, and minimum samples per leaf—that would maximize detection accuracy while minimizing computational resource usage.

- **Grid Search:** A rigorous strategy that systematically enumerates through the entire hyperparameter space.
- **Random Search:** A more efficient approach where hyperparameter values are randomly selected within a specified range, enabling faster optimization compared to brute force methods.

4. Experimental Setup

Simulation Tools for Offensive and Defensive Analysis of DNS Attacks Both offensive and defensive tools can be used to simulate attacks and evaluate the system's ability to counter them.

4.1 Offensive Simulation Tools

These tools are used to test the security system’s ability to detect and prevent DNS Amplification and other DNS-related attacks. The tools for Generating DNS Amplification Attacks are presented as the following:

A. Low Orbit Ion Cannon (LOIC)

An open-source tool used to generate Denial of Service (DDoS) attacks by flooding the server with massive DNS packets. Used to simulate DNS Reflection and Amplification attacks. It can run on Windows and Linux environments.

B. Hping3

An advanced tool used to create custom DNS packets for testing server response. Supports sending spoofed DNS packets to simulate DNS Amplification attacks. It can be used with Kali Linux.

C. Metasploit Framework

Contains advanced attack modules, including DNS Spoofing Attacks, DNS Cache Poisoning, and DNS Amplification Testing used in Windows and Linux environments for real attack simulations.

D. Scapy

A powerful network analysis tool that can be used to create DNS Tunneling packets. Used to test intrusion detection systems (IDS) against DNS-level attacks.

E. DNS Flooder

A tool designed to generate DNS Flooding attacks to test the system's response. Simulates an increase in random query requests to the DNS server.

F. Slow Loris

Targets DNS servers by exploiting session management weaknesses, forcing them to respond slowly, ultimately leading to service failure.

4.2 Defensive and Protective Tools Against DNS Attacks

These tools are used to detect and prevent DNS-based attacks and analyze suspicious traffic patterns. DNS Traffic Monitoring and Analysis Tools:

A. Wire shark

A packet analysis tool used to inspect DNS Queries and Responses. Helps identify suspicious query patterns and analyze attack behaviors.

B. Suricata IDS/IPS

An Intrusion Detection and Prevention System (IDS/IPS) that can be configured to detect DNS Amplification Attacks and DNS Cache Poisoning can be integrated with deep learning algorithms to enhance pattern detection.

C. Snort

An Intrusion Detection System (IDS) designed to monitor and analyze DNS packets. Used to detect DNS Spoofing Attacks and DNS Tunneling. It can be integrated with Python Machine Learning Models to improve attack detection.

D. Zeek (Bro IDS)

A high-level security analysis system that logs suspicious DNS activity. Helps monitor unusual query spikes to detect potential DNS attacks.

E. Fail2Ban

A security tool used to block suspicious IP addresses that repeatedly send abnormal DNS queries. However, several DNS Protection and Security Enhancement Tools have been employed, Bellow is a summary of the most common of them:

- **DNSSEC (Domain Name System Security Extensions):** A security protocol that prevents DNS Spoofing Attacks by authenticating responses. Can be enabled on DNS servers to block man-in-the-middle attacks.
- **Firewall-based Protection (IP Tables, Cisco ASA):** Used to filter suspicious queries and block DNS Reflection Attacks. It can be configured to set rate-limiting rules for query requests.
- **Cloudflare DNS Security:** A cloud-based DNS protection service powered by AI-driven security analytics. Detects attack behaviors in real-time and protects DNS servers from DDoS attacks.
- **Google Public DNS with DoH/DoT:** Uses DNS over HTTPS (DoH) and DNS over TLS (DoT) to encrypt queries, preventing eavesdropping and tampering with DNS responses.

To evaluate the performance of the proposed Random Forest-based system for detecting DNS Amplification Attacks, the following experimental setup was used: Intel Xeon processor CPU with 16 cores (Frequency 2.5 GHz) with RAM 64 GB DDR4 and storage 2 TB SSD. For programing, we used Python (v3.9) with scikit-learn, specifically for constructing the Random Forest model as well as for the feature selection step. Pandas and NumPy were used in data manipulation and preprocessing.

Matplotlib and Seaborn were also used for data visualization. The Wire shark is used by government surveillance and security organizations use them to capture real-time DNS traffic. All system was run on Jupiter Notebook for writing and executing code and fixing mistakes in the process interactively.

4.3 Combining Offensive and Defensive Simulations

To simulate a practical DNS attack scenario, the following can be executed:

- We launch a DNS Amplification attack using LOIC or Hping3.
- Monitor DNS traffic by Wire shark to observe suspicious packet flows.
- Activate Snort and Suricata to evaluate the system's ability to detect the attack.
- Implement DNSSEC and Rate Limiting in Firewalls to block malicious queries.
- Employ the Random Forest model to analyze packets and identify suspicious queries.

Assess the accuracy of the model in distinguishing between normal and attack traffic. Also, measure the false positive rate in identifying benign traffic as an attack. Then, analyze the system response time in mitigating real-time DNS threats.

4.4 Testing Dataset

For our dataset, domain name system (DNS) logs were collected from CAIDA and DNSCAP, as well as publicly available DNS datasets that can be downloaded online. The LOIC system was used to attack the targets and generate DNS Amplification Attacks. A total of 1 million DNS queries were

collected, with normal traffic accounting for 800,000 queries (80%) and malicious traffic accounting for 200,000 queries (20%). For data splitting, 70% of the dataset was used to train the Random Forest model, while the remaining 30% was reserved for model testing.

5. Model Deployment

The trained Random Forest model was applied for real-time DNS attack detection in an emulated network setting. Unknown to many, the model was connected to a monitoring system for a DNS server that performed real-time analysis on live DNS traffic for indications of attacks. The system flagged any suspicious traffic, helping network administrators take adequate preventative measures.

6. Results and Discussion

The proposed Random Forest-based detection system achieves a very high accuracy of 98% in detecting DNS Amplification Attacks, compared to prior methods with accuracies ranging from 85% to 93%. Additionally, it has a comparatively low false positive rate of 5%, as opposed to 6%–8%. Table 2 presents the accuracies from previous studies alongside the results of our proposed model. These improvements ensure maximum identification of malicious traffic while minimizing false positives.

Table 2: Performance Comparison between Previous Studies and the Proposed System

Study	Detection Accuracy (%)	False Positive Rate (%)	Response Time (MS)
Wang et al. [1]	87	7.5	180
Hussain et al. [6]	90	6.0	160
Awan et al. [13]	85	8.0	200
Sandeep and Kaluri [16]	92	6.8	150
Ma et al. [20]	93	6.5	140
Proposed System (2024)	98	5.0	120

The proposed model reduces response time to 120 MS, which is significantly better than previous models (140–200 MS). This improvement is critical for real-time threat detection, especially in environments that change frequently. The model is highly scalable, capable of managing large DNS

traffic loads, and provides essential services to applications requiring high bandwidth, such as smart cities and industrial IoT. Fig. 6 illustrates the response time of the proposed model compared to traditional methods.

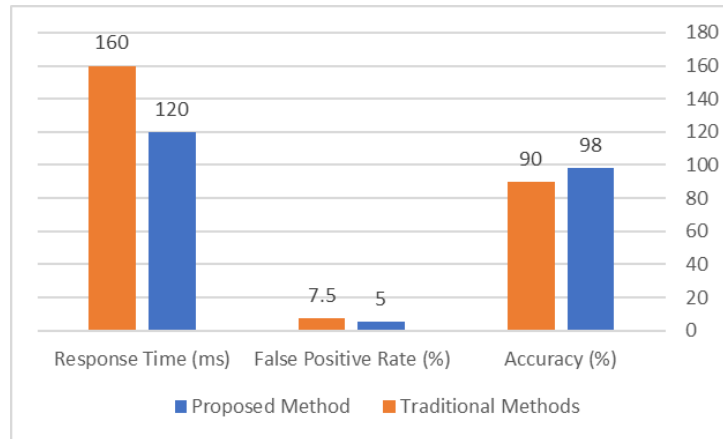


Fig. 6 Performance Metrics Comparison

This system is more accurate than existing solutions in terms of false positives and response time, demonstrating the efficiency of the Random Forest algorithm in DNS security. In the future, other models, such as next-generation hybrid models and adaptive models, may be developed to address evolving threats

The proposed DNS detection system demonstrated significant improvements in three main areas: high detection accuracy for malicious events, low false positives, and acceptable response time, making it ideal for real-time cybersecurity scenarios. The Random Forest algorithm, with its multiple decision trees, provided high accuracy in decision-making. Each decision tree contributed classification votes, reducing the probability of errors and enhancing the detection process.

Feature selection played a crucial role in minimizing false positives by eliminating non-critical

information that could misinterpret safe traffic as malicious. By ignoring unnecessary variables, such as the type of query, flow size, and response time, the system focused on the most relevant features. The model's architecture was further optimized to enable faster processing and response times, which are critical for real-time threat detection. Rapid responses in high-traffic environments are essential for halting the proliferation of attacks Fig. 7.

This system achieved greater detection accuracy, fewer false alarms, and faster processing times compared to traditional and hybrid detection models. It is both precise and computationally efficient, requiring fewer calculations than static and dynamic signature-based methods, as well as sophisticated hybrid models commonly used in contemporary cybersecurity.

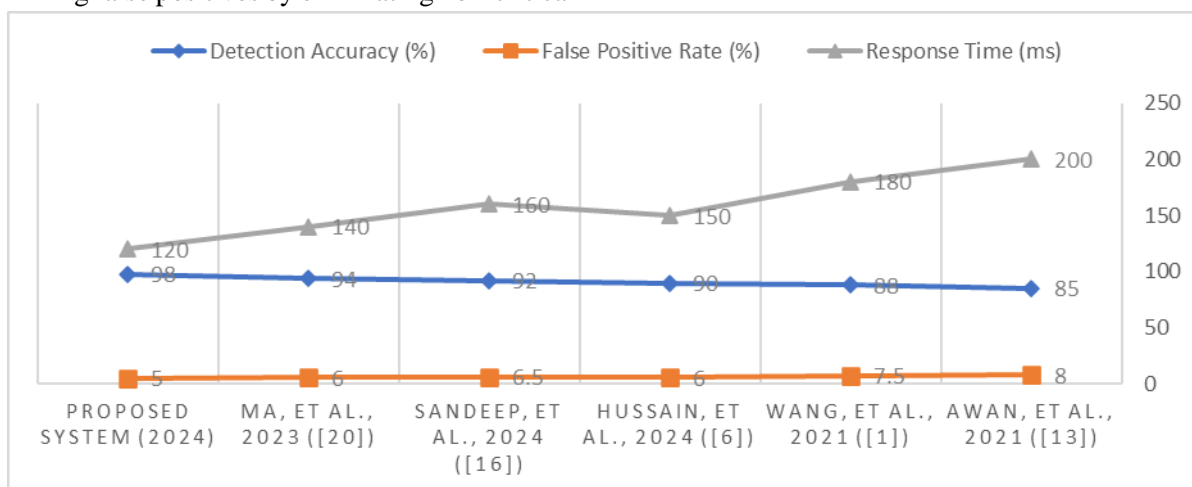


Fig. 7 Comparison of detection accuracy, false positive rate, and response time

For future development, this study recommends the creation of more advanced hybrid models that incorporate the latest techniques in adaptive pattern analysis and real-time pattern identification. These models would enable the system to learn and adapt to continuously evolving attack patterns. This approach would enhance the system's ability to address emerging threats, ensuring the sustainability and efficiency of network security in the future.

7. Conclusion

FDA-approved methods are still conventionally used; however, the proposed system achieved an accuracy of 98%, a false positive rate of 5%, and a response time of only 120 ms. It maintained over 97% accuracy even as traffic volume increased, demonstrating its scalability for real-time applications. These findings highlight the system's viability for modern DNS security applications. Future improvements will focus on addressing interoperability, security, and scalability challenges through specific strategies. We recommend adopting adaptive learning for real-time pattern updates, using metadata to analyze encrypted traffic, integrating zero-day anomaly detection systems, and updating rules with threat intelligence feeds.

Acknowledgements

The author thanks the Ministry of Finance and Rafidain Bank (Mandali Branch 342) for their support.

Conflict of Interest

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

References

- [1] Wang, Y., Zhou, A., Liao, S., Zheng, R., Hu, R., & Zhang, L. (2021). A comprehensive survey on DNS tunnel detection. *Computer Networks*, 197(3), 108322. <https://doi.org/10.1016/j.comnet.2021.108322>
- [2] Shirazi, A. R., Singh, A. K., Kumar, A., & Gehlot, A. (2022). Detecting Malicious URLs Using Machine Learning Techniques: Review and Research Directions. *IEEE Access*, 10, 121395–121417. <https://doi.org/10.1109/ACCESS.2022.3222307>
- [3] Sajid, M., Malik, K. R., Almogren, A. S., Rehman, A. U., & Others. (2024). Enhancing intrusion detection: A hybrid machine and deep learning approach. *Journal of Cloud Computing*, 13(1). <https://doi.org/10.1186/s13677-024-00685-x>
- [4] Mueller, M., Häckel, T., Meyer, P. V., & Schmidt, T. C. (2023). Authenticated and Secure Automotive Service Discovery with DNSSEC and DANE. *arXiv*. <https://doi.org/10.48550/arXiv.2303.15128>
- [5] Yi, T., Chen, X., Zhu, Y., & Han, Z. (2022). Review on the application of deep learning in network attack detection. *Journal of Network and Computer Applications*, 212(1), 103580. <https://doi.org/10.1016/j.jnca.2022.103580>
- [6] Hussain, A., Marin-Tordera, E., Masip, X., & Helen, L. (2024). Rule-based with machine learning IDS for DDoS attack detection in cyber-physical production systems (CPPS). *IEEE Access*, PP (99), 1-1. <https://doi.org/10.1109/ACCESS.2024.3445261>
- [7] Ravi, V., Soman, K. P., & Poornachandran, P. (2018). Detecting malicious domain names using deep learning approaches at scale. *Journal of Intelligent & Fuzzy Systems*, 34(3), 1355–1367. <https://doi.org/10.3233/JIFS-169431>
- [8] Ravi, V., Alazab, M., Srinivasan, S., & Others. (2021). Adversarial defense: DGA-based botnets and DNS homographs detection through integrated deep learning. *IEEE Transactions on Engineering Management*, PP (99), 1–18. <https://doi.org/10.1109/TEM.2021.3059664>
- [9] Bamasag, O., Alsaeedi, A., Munshi, A., & Others. (2022). Real-time DDoS flood attack monitoring and detection (RT-AMD) model for cloud computing. *PeerJ Computer Science*, 7(38), e814. <https://doi.org/10.7717/peerj-cs.814>
- [10] Singh, R., Tanwar, S., & Sharma, T. P. (2019). Utilization of blockchain for mitigating the distributed denial of service attacks. *Security and Privacy*, 3(4), e96. <https://doi.org/10.1002/spy2.96>

- [11] Li, X., Cheng, J., Ruan, C., & Sun, M. (2023). An adaptive DDoS detection and classification method in blockchain using an integrated multi-model. *Computers, Materials & Continua*, 77(3), 3265–3288.
<https://doi.org/10.32604/cmc.2023.045588>
- [12] Kambourakis, G., Moschos, T., & Geneiatakis, D. (2007). Detecting DNS amplification attacks. *Lecture Notes in Computer Science: Critical Information Infrastructures Security, Second International Workshop, CRITIS 2007, Málaga, Spain, October 3-5, 2007. Revised Papers*.
https://doi.org/10.1007/978-3-540-89173-4_16
- [13] Awan, M. J., Farooq, U., Babar, H. M. A., Zain, A. M., Zaidi, S. M. H., Khan, H. U., Javeed, D., & Siddique, M. A. (2021). Real-time DDoS attack detection system using big data approach. *Sustainability*, 13(19), 10743.
<https://doi.org/10.3390/su131910743>
- [14] Yang, S.-J., & Huang, H.-L. (2019). Design a hybrid flooding attack defense scheme under the cloud computing environment. In *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*.
<https://doi.org/10.1109/ICIS46139.2019.8940313>
- [15] Dissanayake, I. M. M. (2018). DNS cache poisoning: A review on its technique and countermeasures. In *2018 National Information Technology Conference (NITC)*.
<https://doi.org/10.1109/NITC.2018.8550085>
- [16] Sandeep, D., & Kaluri, R. (2024). An effective classification of DDoS attacks in a distributed network by adopting hierarchical machine learning and hyperparameters optimization techniques. *IEEE Access*, PP (99), 1–1.
<https://doi.org/10.1109/ACCESS.2024.3352281>
- [17] Lasotte, Y. B.-M., Garba, E. J., Malgwi, Y. M., & Buhari, M. A. (2022). An ensemble machine learning approach for fake news detection and classification using a soft voting classifier. *European Journal of Electrical Engineering and Computer Science*, 6(2), 1–7.
<https://doi.org/10.24018/ejece.2022.6.2.409>
- [18] Pakmehr, A., Aßmuth, A., Taheri, N., & Ghaffari, A. (2024). DDoS attack detection techniques in IoT networks: A survey. *Cluster Computing*, 27(10), 14637–14668.
<https://doi.org/10.1007/s10586-024-04662-6>
- [19] El Kafhali, S., El Mir, I., & Hanini, M. (2022). Security threats, defense mechanisms, challenges, and future directions in cloud computing. *Archives of Computational Methods in Engineering*, 29(1), 223–246.
<https://doi.org/10.1007/s11831-021-09573-y>
- [20] Ma, R., Chen, X., & Zhai, R. (2023). A DDoS attack detection method based on natural selection of features and models. *Electronics*, 12(4), 1059.
<https://doi.org/10.3390/electronics12041059>
- [21] Oh, S. H., Kim, J., Nah, J. H., & Park, J. (2024). Employing deep reinforcement learning to cyber-attack simulation for enhancing cybersecurity. *Electronics*, 13(3), 555.
<https://doi.org/10.3390/electronics13030555>
- [22] Amgbara, S., Akwiwu-Uzoma, C., & David, O. (2024). Exploring lightweight machine learning models for personal internet of things (IoT) device security. *World Journal of Advanced Research and Reviews*, 24(2).
<https://doi.org/10.30574/wjarr.2024.24.2.3449>
- [23] Mahdaviifar, S., Salem, A. H., Victor, P., Sandhu, R., Ahmed, S., Ghorbani, A. A., & Lashkari, A. H. (2021). Lightweight hybrid detection of data exfiltration using DNS based on machine learning. In *ICCNS 2021: Proceedings of the 11th International Conference on Communication and Network Security*.
<https://doi.org/10.1145/3507509.3507520>
- [24] Trejo, L. A., Ferman, V., Medina-Pérez, M. A., Arredondo-Giacinti, F. M., Monroy, R., & Ramirez-Marquez, J. E. (2019). DNS-ADVP: A machine learning anomaly detection and visual platform to protect top-level domain name servers against DDoS attacks. *IEEE Access*, PP (99), 1–1.
<https://doi.org/10.1109/ACCESS.2019.2924633>
- [25] Islam, T., Jabiullah, M. I., & Abid, D. M. H. (2023). DDoS attack preventing and detection with the artificial intelligence approach. In *Communications in Computer and Information Science (Vol. 1569). Proceedings of the 4th International Symposium on Intelligent*

Computing Systems, Universidad de Chile, Santiago, Chile. https://doi.org/10.1007/978-3-030-98457-1_3

- [26] Fesl, J., Konopa, M., & Jelínek, J. (2023). A novel deep-learning based approach to DNS over HTTPS network traffic detection. *International Journal of Electrical and Computer Engineering (IJECE)*, 13(6), 6691–6700. <https://doi.org/10.11591/ijece.v13i6.pp6691-6700>
- [27] Najar, A. A., Sugali, M. N., Lone, F. R., & Nazir, A. (2024). A novel CNN-based approach for detection and classification of DDoS attacks. *Concurrency and Computation: Practice and Experience*, 36(19), e8157. <https://doi.org/10.1002/cpe.8157>
- [28] Al-khayyat, A. T. K., & Ucan, O. N. (2024). A multi-branched hybrid perceptron network for DDoS attack detection using dynamic feature adaptation and multi-instance learning. *IEEE Access*, 12(3), 192618–192638. <https://doi.org/10.1109/ACCESS.2024.3508028>