Computers & Security 157 (2025) 104561

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full length article

NIOM-DGA: Nature-inspired optimised ML-based model for DGA detection

Daniel Jeremiah [a], Husnain Rafiq [a,*], Vinh Thong Ta [b], Muhammad Usman [a], Mohsin Raza [c], Muhammad Awais [c]

[a] Department of Computer Science, Edge Hill University, Ormskirk, UK
[b] Cranfield Defence and Security, Shrivenham, UK
[c] Department of Computer Science, University of East Anglia, Norwich, UK

ARTICLE INFO

Keywords:
Domain Generation Algorithm
Machine Learning
Malware
Nature Inspired Optimisation

ABSTRACT

Domain Generation Algorithms (DGAs) allow malware to evade detection by generating millions of random domains daily for Command-and-Control (C&C) communication, challenging traditional detection methods. This work presents NIOM-DGA, a novel machine learning model that applies nature-inspired algorithms (NIAs) to select an optimal subset of 78 features from a dataset of over 16 million domain names, including several features not traditionally used in DGA detection. This approach enhances accuracy, robustness, and generalisability, achieving up to 98.3% accuracy—outperforming most existing approaches. Further testing on 10 external datasets with over 37 million domains confirms an average classification accuracy of 95.7%. Designed for seamless integration into SIEM, EDR, XDR, and cloud security platforms, NIOM-DGA significantly improves DGA detection compared to existing methods, advancing practical threat detection capabilities.

1. Introduction

In recent years, ransomware has emerged as a significant cybersecurity threat, with a notable increase in global cyber attacks (Tyagi, 2023; Griffiths, 2023; Ford, 2024). Statistics from 2021 reveal a concerning surge, with over 623.3 million attempted ransomware incidents recorded, marking a substantial 105% rise compared to the previous year 2022 (Griffiths, 2023). Despite a slight decrease of 23% in 2022, ransomware attacks remain prevalent, posing persistent challenges to individuals and organisations around the world. Between 2023 and 2024, the global share of users affected by ransomware attacks increased to 0.44%, a 0.02 percentage point rise . The average ransom payment in 2024 reached $2.73 million, nearly $1 million higher than in 2023 . Ransomware damages are expected to hit $265 billion annually by 2031 (Kaspersky, 2025; Chen et al., 2024). DGAs are pivotal in ransomware operations in Command-and-Control (C&C), enabling threat actors to dynamically generate many domain names for both malware agents and the C&C Server. By employing DGAs, cybercriminals can obfuscate their malicious activities, making it challenging for traditional detection mechanisms to identify and mitigate ransomware threats effectively (Kemmerling, 2023; Tuan et al., 2022).

Several malware's families utilises DGAs to locate and connect to C&C servers. This method allows malware creators to evade the domain blacklist of the C&C server. This technique poses challenges for security measures to blacklist all potential domains generated by DGAs. Since threat actors only register a minor portion of these domains, DGA-infected malware generates numerous unresolved DNS queries. Some can lead to a resolved IP that can connect to the C&C server (Anderson et al., 2016; Shahzad et al., 2021). In addition, relying solely on the heuristic blocking method is unreliable for security solutions, as legitimate computing systems may generate unresolved DNS queries for benign purposes. The significance of DGAs extends beyond ransomware, which includes various other forms of malware proliferation. In addition, Botnets, Trojans, information stealers or Remote Access Trojans (RATs), adware, and spyware are among the many types of malware that use DGA to establish communication channels with (C&C) Server (Suryotrisongko and Musashi, 2022; Sun and Liu, 2023). Botnets, for instance, utilise DGAs to facilitate coordination among compromised devices, enabling attackers to orchestrate large-scale distributed denial-of-service (DDoS) attacks or propagate additional malware (Sea and Law, 2023; Ding et al., 2023). Trojans and RATs employ DGAs to maintain stealthy communication with command and control servers, allowing threat actors to remotely control compromised systems and exfiltrate sensitive information (Weissgerber et al., 2023). Similarly, adware and spyware employ DGAs to evade detection and persistently monitor and collect user data for malicious purposes.

Traditional methods such as domain blocklists, DNS sinkholing, IDS rules, and IP-based blacklists, fall short in effectively mitigating DGA-based threats (Zhao et al., 2023a). For instance, domain blocklists

* Corresponding author.
E-mail address: rafiqh@edgehill.ac.uk (H. Rafiq).

https://doi.org/10.1016/j.cose.2025.104561
Received 17 September 2024; Received in revised form 20 May 2025; Accepted 31 May 2025
Available online 21 June 2025

can be easily circumvented as DGAs are capable of generating over 50,000 domains daily, rendering static blacklists obsolete almost immediately (Highnam et al., 2021; Sun and Liu, 2023). DNS sinkholing and rule-based IDS systems also struggle with the high rate of false positives, often exceeding 20%–30%, due to benign domains occasionally exhibiting similar entropy or structure to malicious ones (Highnam et al., 2021; Sun and Liu, 2023).

In addition, DNS sinkholing which redirects traffic from malicious domains to controlled servers, allowing organisations to intercept communications intended for DGA-generated domains and disrupt malicious activity (Quezada et al., 2022). While effective in theory, DNS Sinkholing requires significant expertise and resources to set up and maintain sinkhole servers. The above-cited reactive measures rely on identifying and redirecting traffic to known malicious domains, making them less effective against unknown DGA-generated domains. Another traditional method employ IDS rule-based systems, which rely on predefined rules to detect and block network traffic associated with DGA activity (Zhao et al., 2023a). However, defining accurate and exhaustive rules to cover all possible DGA patterns is increasingly difficult as DGA domains are newly generated based on different algorithm patterns. Due to the dynamic nature of DGA, IDS rule-based approaches may generate high false positive rates or miss emerging DGA threats altogether. IP-based blacklists block access to known malicious IP addresses associated with DGA activity. While this approach can effectively block communication with malicious C2 servers, it does not address the root cause of DGA-based threats—the dynamic generation of domain names. Consequently, attackers can quickly switch to new IP addresses, rendering IP-based blacklists less effective in the long term.

Introducing Machine Learning (ML) to the challenge of detecting and mitigating DGA activity offers promising results (Kostopoulos et al., 2023; Givre; Quezada et al., 2022). ML algorithms can analyse patterns in network traffic data to identify anomalous behaviours associated with DGA-generated domain names. ML models can effectively differentiate between legitimate domain traffic and potentially malicious DGA activity by training on historical data and learning from diverse features extracted from DGA domains. ML algorithms, such as Random Forest (RF), Decision Trees (DT), ExtraTrees, Logistic Regression (LR), XGBoost (XGB), AdaBoostClassifier, and Multi-layer Perceptron (MLP) Classifier, can be trained on labelled datasets to recognise patterns indicative of DGA traffic (Kostopoulos et al., 2023). These algorithms can then be deployed in real-time to analyse DNS traffic, automatically flagging and blocking connections to suspicious domain names generated by DGAs. Many researchers have developed ML models to detect and prevent malware using DGA techniques (Sreekanta, 2022; Ben, 2024; Velasco-Mata et al., 2023).

However, a recent attack from Lockbit shows that such DGA attacks are still effective due to the lack of a feature engineering process and the nature of the dataset used to train the ML model (Tyagi, 2023). Furthermore, most ML-based DGA detection models in prior research typically rely on limited datasets (often under 1 million samples), restricting their ability to generalise across diverse malware families. NIOM-DGA (Nature-Inspired Optimised Machine Learning model for DGA detection) addresses the dynamic and evasive characteristics of domain generation algorithms by leveraging a robust feature engineering pipeline. This pipeline integrates 78 algorithmic and statistical domain name features, including entropy measures, sequence alignment metrics (e.g., Levenshtein, Jaro, Needleman–Wunsch), and vector-based similarity measures (e.g., cosine similarity). By capturing complex lexical and structural patterns in domain names, NIOM-DGA achieves improved detection accuracy over prior models that fail to generalise across diverse DGA families.

Furthermore, our feature extraction process integrates alphabetic combinations, ensuring a comprehensive representation of domain characteristics. This methodology has enabled our model to achieve high levels of accuracy and robustness. Nevertheless, Kostopoulos et al. (2023) emphasise that even with reported high accuracies of traditional ML, specific DGA malware variants may still evade detection if the ML features extraction process is inadequate (Kostopoulos et al., 2023; Quezada et al., 2022; Gogoi and Ahmed, 2023).

Conversely, we further improved accuracy by integrating Nature-Inspired Algorithms (NIAs) for hyperparameter tuning, specifically utilising the Bat Algorithm (BA), Firefly Algorithm (FA), and Grey Wolf Optimiser (GWO). The hyperparameter configurations included **n_estimators** set to 60 for BA and 80 for both FA and GWO, and **max_depth** set to 34 for BA and 28 for FA and GWO. Additionally, the **min_samples_split** parameter was consistently set to 2 across all algorithms, while the **max_features** parameter varied, being set to *auto* for BA and GWO and *sqrt* for FA. The results demonstrate that achieving high accuracy is closely tied to hyperparameter tuning, as detailed in our findings.

In this work, we propose a NIOM-DGA model; NIOM-DGA is trained on 16 million distinct domain names comprising DGAs-generated and benign samples. Furthermore, NIOM-DGA utilises eleven (11) different datasets to evaluate the performance of the proposed model extensively. To further enhance the performance of our model, we utilised advanced nature-inspired algorithm optimisation techniques (Pye et al., 2020; Camacho Villalón et al., 2020). Specifically, algorithms such as Bat, Firefly, and Grey Wolf Optimiser were employed to fine-tune the hyperparameters of the best-performing classifier. This detailed optimisation aimed to ensure optimal performance and generalisability of the model across various DGA families, enhancing its effectiveness in real-world scenarios (Pye et al., 2020; Camacho Villalón et al., 2020). The key contributions of this work can be summarised as follows:

1. NIOM-DGA integrates 78 sophisticated domain characteristics-algorithmic and advanced feature extraction processes that have not been previously observed in the literature (to the best of our knowledge). This novel integration of features significantly improved the detection capability for unknown DGAs.
2. We propose NIOM-DGA, a Nature-inspired Optimised ML-based novel model for DGA detection. The proposed model is trained on a balanced DGA-generated and benign domains dataset and optimised using nature-inspired algorithms to enhance detection performance. The NIOM-DGA model significantly outperforms various DGA detection models proposed in the literature.
3. To the best of our knowledge, we have employed the most comprehensive dataset to date, comprising over 16 million benign and dga domain names, to train the model and we evaluate the model using ten different external datasets to assess the generalisation capability of the proposed model.
4. We publish an updated version of the dataset derived from the ExtraHop Network dataset used in this research. We cleaned the original dataset and ensured that there were no duplicates, removed and cleaned the data suitable for our feature selection method, and also cleaned all the 37 million external domain names used for this research.

The subsequent sections of the paper are structured as follows. Section 2 offers a background on the research topic. In Section 3, we explore Related Work, examining methodologies from previous studies, and identifying gaps in the literature. Section 4 presents an overview of the Extrahop Network Dataset. Section 5 introduces NIOM-DGA, our innovative framework for DGA detection, elucidating its architecture, feature engineering process, and optimisation using nature-inspired algorithms, and detailed experimental outcomes analysis. Section 6 conducts a Comparative Analysis with other approaches. Section 7 concludes with a Conclusion and outlines Future Work. Finally, Section 8 provides a comprehensive list of abbreviations used throughout the paper to ensure clarity and consistency in terminology.

2. Background

DGAs are a significant challenge in cybersecurity due to their role in malware propagation and command-and-control (C2) infrastructure (Tuan et al., 2022; Suryotrisongko and Musashi, 2022). Malicious actors dynamically use DGAs to generate domain names, enabling communication between malware-infected devices and remote servers (Tuan et al., 2022; Suryotrisongko and Musashi, 2022). Unlike static domains, DGAs create domain names dynamically, making it difficult for security solutions to detect and block malicious traffic (Pan et al., 2022; Quezada et al., 2022). DGAs use algorithms and seed values to generate domain names, often incorporating pseudo-randomness and specific patterns (Sreekanta, 2022; Velasco-Mata et al., 2023). This method allows the malware to establish communication channels with C2 servers without relying on fixed addresses that can be easily blocked (Kostopoulos et al., 2023; Sidi et al., 2019). Over time, DGAs have evolved to include more sophisticated techniques, such as using Generative Adversarial Networks (GANs), which generate domain names that closely resemble legitimate ones (Sidi et al., 2019; Ren et al., 2023; Corley et al., 2020).

Additionally, DNS fast flux is another technique used for malicious purposes, involving rapid changes to the IP addresses associated with domain names (Rana and Aksoy, 2021). Attackers use DNS fast flux to conceal the actual location of malicious servers and evade detection by security solutions. By constantly changing the mapping between domain names and IP addresses, attackers make it challenging for security detection's to pinpoint and block malicious infrastructure (Rana and Aksoy, 2021). While DNS fast flux can be effective in evading detection and blocking by security solutions, it also has limitations that make it less effective than DGAs in specific scenarios (Katz, 2021; Rana and Aksoy, 2021). One limitation of DNS fast flux is its reliance on a central server or set of servers to redirect traffic to the constantly changing IP addresses. This centralisation creates a single point of failure that security detection's can target and disrupt. Additionally, the rapid changes in IP addresses associated with DNS fast flux can increase network traffic and latency, potentially raising suspicions and drawing attention to the malicious activity. In contrast, malware that uses DGAs generate domain names dynamically, allowing malware to establish communication channels with command-and-control (C2) servers without relying on a centralised infrastructure like DNS fast flux. This decentralised approach makes DGAs more resilient to takedown efforts by security detection's (Katz, 2021; Rana and Aksoy, 2021).

Moreover, another limitation of DNS's fast flux is its susceptibility to detection by security solutions that monitor DNS traffic patterns. The rapid changes in IP addresses associated with fast flux can trigger alerts and anomalies in DNS logs, leading to increased scrutiny by security detection's (Katz, 2021). In comparison, DGAs operate at the domain name level, making them less susceptible to detection based solely on network traffic patterns. While DNS fast flux can be effective in specific scenarios, its limitations make it less suitable for long-term, persistent communication between malware-infected devices and C2 servers. In addition, DGAs provide a more robust and resilient method for establishing covert communication channels, making them a preferred choice for many malware authors and threat actors (Quezada et al., 2022; Velasco-Mata et al., 2023; Patil et al., 2022). Section 2.1.1 examines three malware families and the algorithmic techniques for generating DGA domain names: Orchard v3, GameoverZeus and the BumbleBee Malware Family.

2.1. Related malware utilising DGA

Extensive research has been conducted on DGA detection by different researchers (Pan et al., 2022; Ren et al., 2023; Kostopoulos et al., 2023; Ahmed et al., 2022). However, limited attention has been paid to the technical aspect of how attackers generate these domain names. This section provided a brief background about DGA malware families and the techniques they use. Table 1 provides a concise overview of various malware families and their respective techniques. We select three specific malware instances for detailed case studies.

2.1.1. Orchard v3 malware

Orchard v3 malware distinguishes itself by its unique Domain Generation Algorithm (DGA), which dynamically generates domain names that the malware uses for communication (Daji and Suqitian, 2022). The DGA is designed to create domains based on two different seed sources: ***"the current date"*** and a ***"blockchain-based value"***. This dual-seeding approach allows Orchard v3 to generate a diverse set of domain names, making it more challenging for security detection to block all potential communication channels. The algorithm begins by accepting a ***date input***, which could be the ***current date*** or any other specified date. This ***date*** is deterministic, meaning it is fixed and predictable once chosen (Daji and Suqitian, 2022). The algorithm is set up to run twice in a loop. On the first iteration, the DGA takes the ***current date*** as input and formats it as a string in the ***"Year-Month-Day"*** format (for example, "***2024-09-06")*** (Bader, 2024). It then appends the domain ***"ojena.duckdns.org"*** to this formatted date, creating a "seed" string. The seed string for this iteration might look like ***"2024-09-06ojena.duckdns.org""***.

On the second iteration, if the 'blockchain' option is enabled, the DGA retrieves a blockchain-based seed. This seed is generated by a function that accesses blockchain data, possibly the Bitcoin Genesis Block. The retrieved blockchain seed is used directly in the same manner as the date seed. Both seeds (the date-based seed and the blockchain-based seed) are then hashed using the MD5 cryptographic hash function. This hashing step produces a 32-character hexadecimal string from each seed, which serves as a base for generating the second-level domains (SLDs) (Bader, 2024). The MD5 output is divided into four segments, each consisting of 8 characters. For example, if the MD5 hash is ***"1a2b3c4d5e6f7g8h9i0jklmnopqrstuv"***, it will be divided into ***""1a2b3c4d", "5e6f7g8h", "9i0jklmn", and "opqrstuv"***. These segments are used as the SLDs in the domain names. In addition, the algorithm pairs these second-level domains with a set of predefined top-level domains (TLDs), which are ***"".com", ".net", ".org", and ".duckdns.org"***. The pairing is achieved through a Cartesian product of the SLDs and TLDs, effectively generating all possible combinations of SLDs and TLDs. Thus, for each 8-character segment of the MD5 hash, four different domains are created, one for each TLD. Given four SLDs from the MD5 hash, the total number of domains generated per run of the algorithm is ***16*** (Bader, 2022). However, since the algorithm runs twice – once for the date-based seed and potentially once for the blockchain-based seed – it can produce up to ***32*** unique domains each day, as shown in Fig. 1.

This design ensures that new domains are generated daily (based on the current date) and potentially also from indeterministic blockchain data, making it harder for security detection's to predict or preemptively block the malware's communication endpoints (Bader, 2022). Moreover, the DGA produces a new set of domains every day with no delay or waiting time between the generation of each domain, maintaining the malware's ability to adapt to changing detection efforts. The use of hexadecimal characters for the SLDs, combined with the diverse range of TLDs, further enhances the unpredictability and variability of the domains generated by Orchard v3. The overall domain format generated by the DGA conforms to a specific structure (Daji and Suqitian, 2022). The second-level domain (SLD) is an 8-character string derived from the MD5 hash, containing characters ranging from ***0-9*** and ***a-f*** (hexadecimal format). This SLD is followed by one of the TLDs mentioned earlier. For example, a domain generated by the algorithm might look like ***1a2b3c4d.com"***" or ***"5e6f7g8 h.duckdns.org"***. The domain names generated exhibit a sequential property, as they follow a fixed generation scheme based on deterministic inputs, making them predictable only to those who understand the algorithm's seeding logic. Orchard v3's DGA uses both deterministic ***(time-based)*** and indeterministic ***((blockchain-based))*** seeds to generate a diverse range of domains, employing MD5 hashing and a combination of predefined TLDs. This approach makes the malware more resilient against defencive measures by continually adapting its communication channels, ensuring that it remains challenging to detect and block consistently over time.

Table 1
Overview of malware Using DGA techniques.

Technique	Description	Advantages	Disadvantages	Malware family (Ref)
Random Generation	Employs a pseudo-random number generator to create seemingly unpredictable domain names.	High domain variability, making prediction very challenging.	Computationally expensive and might result in nonsensical domains.	BumbleBee (Bader, 2023)
Seed-Based Generation	Utilises a seed value (e.g., system information, date/time) as input for a deterministic algorithm to generate domains.	Offers a balance between randomness and predictability for rendezvous.	Vulnerable if the seed selection pattern or algorithm is discovered.	GameoverZeus (Hu et al., 2023; Bader, 2023; Avertium, 2022), Conficker (Wang et al., 2023)
Algorithmic Generation	Leverages a specific algorithm (e.g., cryptographic hash functions, linguistic rules) to produce domains based on a seed or internal state.	Creates a large and potentially unpredictable domain space.	Relies on the chosen algorithm, which could be identified or exploited.	Dridex (Elizarov and Katkov, 2023)
Dictionary-Based Generation	Iterates through a predefined wordlist or dictionary, potentially with modifications, to construct domains.	Large pool of potential domains, making blacklisting less effective.	Relies on a static wordlist, potentially leading to domain exhaustion and predictability.	Mirai (Affinito et al., 2023)
Time-based DGA	Utilises the current date, time, or a combination as seed for the algorithm. Generates new domains at predictable intervals.	Easy to implement, predictable domain generation for rendezvous.	Prone to detection if the seed selection pattern is identified.	Storm Worm (Rao et al., 2023), Orchard v3 (Daji and Suqitian, 2022)
Enumeration-based DGA	Iterates through a predefined wordlist or dictionary to generate domains. May use obfuscation techniques.	Large pool of potential domains, making blacklisting less effective.	Relies on a static wordlist, potentially leading to domain exhaustion.	TrickBot (Papadogiannaki and Ioannidis, 2023; Ding et al., 2023)
Cryptographic Hash Function-based DGA	Employs cryptographic hash functions like MD5 or SHA-1 with a seed (e.g., system information) to generate seemingly random domains.	High variability in generated domains, making prediction difficult.	Computationally expensive for resource-constrained malware.	Locky (Prasetya et al., 2023)
Linguistic-based DGA	Leverages language rules to construct domains. May combine dictionary words with special characters or mutations.	Creates seemingly legitimate domains, potentially evading basic detection.	Can be computationally complex and might result in nonsensical domains.	N/A (Zhao et al., 2023b; Wang et al., 2024)
Multi-level DGA	Combines multiple techniques like time-based and dictionary-based approaches for enhanced complexity.	Offers a wider domain space and makes prediction even harder.	Increases complexity and potential for malfunction within the malware.	Reaper (Sutheekshan et al., 2024)
Context-Aware DGA	Adapts domain generation based on infected system characteristics (e.g., language, location) for better camouflage.	Generates domains that appear more legitimate in specific contexts.	Increases complexity and requires additional information gathering by the malware.	Spargo (Prasetya et al., 2023)
Polymorphic DGA	Dynamically modifies the DGA algorithm itself at runtime to hinder analysis and signature-based detection.	Makes the DGA harder to reverse engineer and detect.	Increases complexity and potential for bugs that disrupt domain generation.	Waledac (Javaheri et al., 2023)

2.1.2. GameoverZeus

The GameoverZeus malware employs a sophisticated Domain Generation Algorithm (DGA) to produce a series of domain names. This process begins with a hashing function, which plays a central role in creating unique and unpredictable domain names (Ashley and Hindarto, 2015). Specifically, the malware uses the MD5 hashing algorithm to convert various pieces of data into a fixed-size string. The hasher function, which is implemented in the code, takes an input string and applies MD5 to produce a 128-bit hash. This hash is represented as a hexadecimal string and serves as the foundation for generating domain names. To further understand this, imagine that each domain name is built on a unique hash value. The hasher function ensures that different inputs yield different hashes, creating a diverse set of possible domain names. This is critical for malware operations, as it helps avoid detection and blocking by security systems (Bader, 2024).

The next step in the process involves extracting the current date using the ***((getDate))*** function. This function breaks down the date into day, month, and year. The date is formatted as ***((YYYY-MM-DD))***, and then split into its components: day, month, and year. This decomposition ensures that the domains generated are tied to specific dates, adding an additional layer of variability. With the date information at hand, the malware proceeds to the seeder function, which generates a seed value essential for the domain name generation process (Ashley and Hindarto, 2015). This function combines a fixed 'salt' value with an index that changes with each iteration. The salt is a constant value that remains the same across all iterations, while the index is unique for each run. The seeder function calculates a remainder when the combined salt and index are divided by 1000 (Bader, 2022). It also performs integer division to get a quotient. These calculations are essential for producing a hash that incorporates both static and dynamic elements. The seed value, derived from these calculations, is then used to generate a hash that is composed of several parts, including the current year and month. This hash is created using the MD5 algorithm, and its value is crucial for generating domain names (Ashley and Hindarto, 2015).

The ***generateDomain*** function is responsible for converting this hash into a domain name. It does this by repeatedly dividing the hash value by ***36*** and using the remainders to select characters. Characters are chosen based on whether the remainder is less than ***10 (digits)*** or ***10*** and above (letters). This process continues until the entire hash

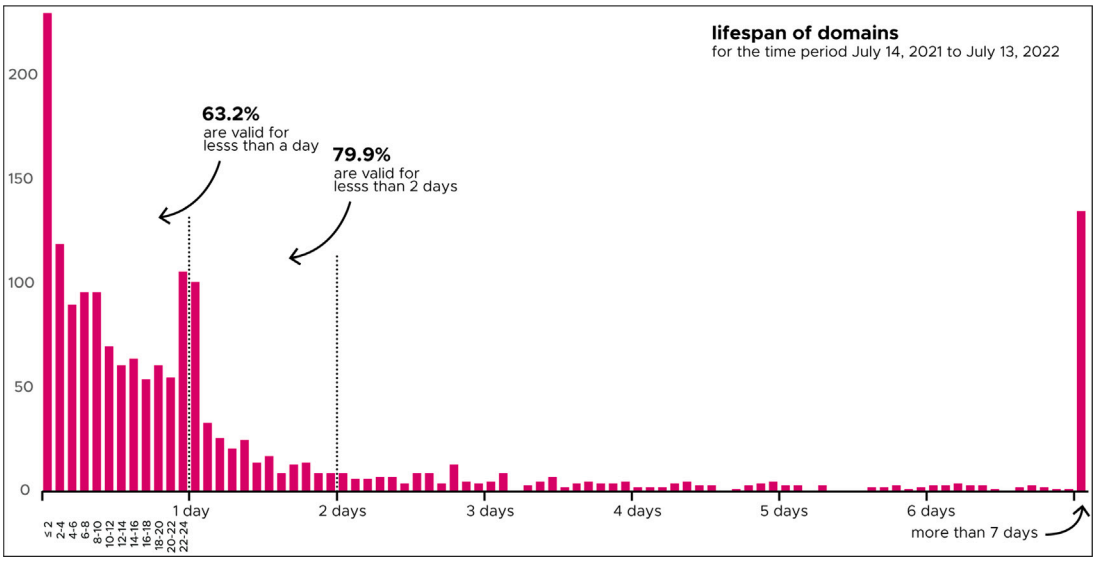


Fig. 1. Lifespan of DGA domains seeded observed in Orchard v3 malware (Bader, 2022).

value has been processed into a string of characters. In addition, to finalise the domain name, the resulting string is reversed, ensuring that each domain name is unique and less predictable. The engine function orchestrates the entire domain generation process. It runs through a specified number of iterations, each time calling the seeder function to produce a new hash based on the current index and salt. This hash is then split into four segments, each of which is used to generate part of the domain name through the ***generateDomain*** function. After generating the domain name, the function appends a suffix based on the value of a derived variable. This suffix can be one of several options, such as ***".com", ".net", or ".biz"***, depending on the calculations involving the ***edx*** value. Finally, the generated domain names are collected into a list and saved to a file. The malware can use this list of domains for its operations, such as connecting to command-and-control servers. The GameoverZeus malware's DGA combines ***hashing, date information, and modular arithmetic*** to create a wide array of domain names. This approach ensures that each domain name is unique, making it difficult for security systems to predict and block them. By understanding this process, NIOM-DGA was explicitly built to detect such malware (Bader, 2022).

2.1.3. BumbleBee malware family

The BumbleBee Malware Family is recognised for its distribution method, which involves an ISO file containing a custom loader dynamic-link library (DLL). Additionally, it has been identified by its unique user agent, "bumblebee" (Bader, 2023; Avertium, 2022). Upon opening the attachment, the malware executes on the victim's system, utilising asynchronous procedure call (APC) injection to run the shellcode received from the command and control (C2) server (Bader, 2023). Bumblebee's primary objective includes downloading and executing additional payloads like shellcode, Cobalt Strike, Silver, and Meterpreter (Bader, 2023; Avertium, 2022). In addition, Bumblebee employs a DGA to generate domain names for communication with the C2 server (Bader, 2023; Avertium, 2022). In March 2022, Bumblebee was observed being used by three threat groups in a DocuSign-branded email campaign, wherein victims were misled into downloading malicious ISO files via email hyperlinks (Bader, 2023). The DGA function employed by Bumblebee is written in Python code. It calculates the seed value and generates domain names iteratively based on this seed value. The DGA function generates domain names comprising 11 characters randomly selected from a character set, followed by the suffix ".life".

BumbleBee DGA Implementation

The DGA employed by BumbleBee generates domain names utilising a Linear Congruential Generator ***LCG***. The ***LCG*** employs standard parameters, with a multiplier of 1664525 and an increment of 1013904223. This generator produces random numbers based on a given seed value, which can be time-dependent or time-independent. If time-dependent, the seed changes based on the current year, month, and second, while a fixed magic seed value is utilised otherwise (Bader, 2023). The seed generation function incorporates both time-based and fixed components. The function calculates the seed value for time-dependent seeds using the current time's year, month, and second. Conversely, a predetermined magic seed value is utilised for fixed seeds. The random number generator ***RNG*** function operates by iteratively applying the ***LCG*** formula to the seed value. This formula produces a sequence of pseudo-random numbers subsequently used to generate the domain names. The domain generation function (DGA) generates domain names by selecting characters from a predefined character set and appending the suffix ".life" to them. The DGA iteratively applies the ***RNG*** function to produce a series of domain names (Bader, 2023).

Seed Generation Function

The ***seed*** function generates the initial seed value the ***RNG*** uses. It takes two parameters: ***magic***, a fixed value used for time-independent seeds, and ***time***, an optional parameter representing the current time. If the ***time*** parameter is provided, the function calculates the seed value based on the current second, month (minus 1 to adjust to the range 0–11), and year. Otherwise, it uses default values for these components.

Random Number Generator (RNG) Function

The ***rand*** function implements a Linear Congruential Generator ***LCG***, a type of pseudo-random number generator. It takes an initial seed value ***r*** and iteratively applies a specific formula to produce a sequence of pseudo-random numbers. The formula involves multiplication by a constant multiplier ***1664525*** and the addition of a continuous increment ***1013904223***. These constants provide desirable statistical properties to the generated random numbers.

Domain Generation Function

The ***dga*** function utilises the random number generator ***rand*** to generate domain names. It iteratively generates characters for the domain name using a predefined character set ***charset***. The function generates a pseudo-random number for each character using the ***rand*** function and selects a character from the ***charset*** based on this number.

It repeats this process for each character in the domain name and appends the suffix ***.life*** to form the complete domain name.

Functionality

The code first generates an initial seed value using the ***seed*** function. The ***rand*** function then uses this seed value to produce a sequence of pseudo-random numbers. The ***dga*** function uses these random numbers to generate domain names. The final output is a series of domain names with the suffix ***.life*** generated based on the given seed value. Understanding the methods and algorithms employed by BumbleBee can aid in research for feature extraction processes aimed at detecting similar domain names. BumbleBee utilises a deterministic algorithm to generate domain names based on a seed value, resulting in seemingly random yet predictable outcomes.

3. Related work

The Domain Name System (DNS) is vital for communication over the internet, translating user-friendly domain names into IP addresses [2]. DGAs generate Second-Level Domain (SLD) labels due to limited top-level domain (TLD). A top-level domain (TLD) with Internationalised Domain Names (IDNs) allows international domain registration from all over the world (Pan et al., 2022). Due to limited TLD usage, threat actors leverage a second-level domain name to generate a Domain generation algorithm (DGA). DGA are the way forward for cybercriminals, as they dynamically generate domain names to evade detection and control the malware Command and control server (C2) (Sidi et al., 2019; Pan et al., 2022; Ren et al., 2023; Kostopoulos et al., 2023). Malware leveraging DGA techniques can create more than 1 million domains daily, complicating threat detection (Kostopoulos et al., 2023; Quezada et al., 2022).

These DGA-generated domains, such as " tbvbnnkkbvsabc1239.ku", serve as decoys for communication with Command Control servers (C2) (Pan et al., 2022; Sidi et al., 2019; Kostopoulos et al., 2023). This type of malware utilises dynamically generated DGA domain names to communicate with command and control servers and can effectively evade detection by frequently switching between domains (Bader, 2022, 2024). This dynamic tactic challenges traditional security measures, hindering threat mitigation (Velasco-Mata et al., 2023; Patil et al., 2022). In addition, these DGA techniques let bots generate DNS requests based on a predetermined seeding mechanism known to the C&C servers. A small set of domain names are registered and expected to be requested for resolution by the bots (Ren et al., 2023; Ben, 2024). These domain names correspond to valid IP addresses of command and control servers, enabling the bots to locate and connect to them. The bots usually send many requests to the DNS (Domain Name System), leading to some requests being successfully resolved while others result in invalid domain names (Corley et al., 2020; Kostopoulos et al., 2023). When a domain name is invalid, the DNS server responds with an "NXDOMAIN" message, indicating that the domain does not exist. Despite most requests yielding no responses, a limited number of DGA domain names are resolved to the C&C IP addresses (Pan et al., 2022; Kostopoulos et al., 2023; Gogoi and Ahmed, 2023).

The substantial volume of DGA domain name queries and the regular changes to the seeding mechanism pose a significant challenge to traditional domain name blocklisting methods like SURBL, Spamhaus DBL, and Malware Domain Blocklist (Brandstaetter, 2024). These conventional methods rely on blocklists containing domain names linked to various threats, including spam, phishing, malware distribution, and botnet command and control servers. In contrast, DGA tactics involve constant domain name hopping and alterations, making these traditional methods ineffective in adequately blocking the DGA technique (Hu et al., 2022).

In the ever-evolving landscape of the last ten years, researchers have extensively employed Machine Learning (ML) techniques to address the challenge of DGAs (Quezada et al., 2022; Shahzad et al., 2021). Machine Learning offers several advantages over traditional methods, such as domain name blocklisting, including SURBL, Spamhaus DBL, and Malware Domain Blocklist. Firstly, ML techniques can adapt and evolve. Unlike static blocklists that rely on predefined criteria to identify malicious domains, ML models can continuously learn from new data and update their algorithms to detect emerging threats (Highnam et al., 2021; Kostopoulos et al., 2023). This adaptability is crucial in the ever-changing landscape of cyber threats, where new malware variants and evasion techniques constantly emerge (Randhawa et al., 2023). Secondly, ML-based approaches can analyse large volumes of data and identify complex patterns that may need to be apparent to human analysts. In addition, ML models can accurately distinguish between benign and malicious activities by leveraging features extracted from vast datasets containing benign and malicious domain names, even when adversaries employ sophisticated evasion tactics (Kostopoulos et al., 2023; Yan et al., 2023). Moreover, ML algorithms can offer a more holistic approach to threat detection by considering multiple features and indicators of malicious behaviour. Cutting-edge research in this field has been undertaken by Ding et al. (2023), Shahzad et al. (2021), Velasco-Mata et al. (2023), Bader (2024), Kostopoulos et al. (2023), wherein researchers have developed a range of machine learning classifiers proficient in identifying DGA domains. These achievements are demonstrated through notable discoveries documented in studies like those by Kostopoulos et al. (2023), Javed et al. (2023), Tuan et al. (2022), Randhawa et al. (2021).

Nevertheless, the efficiency of this ML DGA detection is paramount. One key importance in ML DGA detection is feature extraction and dataset significance (Kostopoulos et al., 2023). Several machine learning algorithms, such as CNN, Bi-LSTM, Random Forest, LSTM, Decision Trees, SVM, Naive Bayes, kNN, and Gradient Boosting, were utilised in a machine learning research project aimed at classifying Domain Generation Algorithm (DGA) domains (Kostopoulos et al., 2023; Javed et al., 2023; Pan et al., 2022). The experiment carried out by the majority of these papers (Pan et al., 2022; Gogoi and Ahmed, 2023; Quezada et al., 2022; Sreekanta, 2022; Velasco-Mata et al., 2023) highlighted several gaps, including a constrained dataset with only a 0.1% representation of DGA domains, the absence of external experimental datasets for evaluating proposed models, and limited features with inadequate feature engineering.

A recent experiment by Kostopoulos et al. (2023) was worthwhile leveraging 50 features during the features engineering process compared to the limited features utilised by Pan et al. (2022), Gogoi and Ahmed (2023). Furthermore, Kostopoulos et al. (2023) investigated the integration of Machine Learning with SHapley Additive exPlanation (SHAP) for enhancing model interpretability. However, the model's effectiveness has limitations in feature extraction and the use of small sample datasets. In the experiment, Adaptive Boosting (AdaBoost) achieved the lowest accuracy at 92.32%, while eXtreme Gradient Boosting (XGBoost) obtained the highest accuracy at 94.81%. Another study by Bronjon Gogoi and Ahmed, Gogoi and Ahmed (2023) yielded remarkable findings with an accuracy rate of 99%. Following a single epoch, the model achieved impressive accuracy. According to the experiment, early stopping techniques were employed to mitigate the risk of overfitting the model. However, the feature extraction process in the Bronjon Gogoi and Ahmed (Gogoi and Ahmed, 2023) experiment is limited due to the leveraged feature vectorisation method. They have tokenisation google.co.in into Unicode characters; both (Kostopoulos et al., 2023; Randhawa et al., 2021; Nowroozi et al., 2022) disagree that top-level domains (TLD) are not necessary during the feature extraction process since domain generation algorithms do not generate the top-level-domains names.

Given the challenges identified in existing state-of-the-art papers on DGA and the limitations of traditional methods such as blocklisting, the ML approach is needed. Another limitation of ML DGA detection is the Adversarial techniques, particularly those employing Generative Adversarial Networks (GANs), have significantly impacted

the effectiveness of Machine Learning (ML) classifiers in detecting Domain Generation Algorithm (DGA) domains (Ren et al., 2023; Corley et al., 2020). GANs are trained on DGA domain datasets and generate synthetic domains that closely resemble real domain names, posing a substantial challenge to traditional classifiers (Sidi et al., 2019; Corley et al., 2020). For example, MaskGAN, utilising characters like underscores and dashes evasion tactics, successfully evades previous classifiers by decreasing the detection from 0.977% to 0.495% by synthesising DGA domains (Sidi et al., 2019). Despite high accuracy in generating look-alike domains, GAN-generated domains may still be detectable by advanced models focusing on domain string characteristics with algorithm capabilities, such as the proposed NIOM-DGA model.

3.1. Challenges in feature extraction process

Machine learning algorithms have shown promise in addressing Malware that takes advantage of DGA-generated domain names by learning patterns and features indicative of DGA-generated domains (Kostopoulos et al., 2023; Ding et al., 2023). By training machine learning classifiers on labelled datasets containing benign and malicious domain names, researchers can develop models capable of accurately classifying benign and data-generated domain names (Highnam et al., 2021; Sivaguru et al., 2020). Features such as domain length, character distribution, and entropy have proven to be valuable indicators of DGA activity (Highnam et al., 2021; Moşolea and Oprişa, 2023; Javed et al., 2023).

However, the effectiveness of machine learning-based DGA detection methods is lacking in terms of the quality and diversity of the training data (N. et al., 2022). Malware authors constantly evolve their techniques to evade detection, requiring ongoing updates and improvements to detection algorithms. While many studies emphasise extracting features from Top-Level Domains (TLDs), this process includes tokenising the dataset using n-grams. The domains and their TLDs are also vectorised and fed into the machine-learning training process (Velasco-Mata et al., 2023; Patsakis and Casino, 2021; Wang and Guo, 2021). Contrary to these findings, Kostopoulos et al. (2023) explain that this approach proves ineffective in detecting DGA attacks. They further highlight that machine learning may struggle to identify DGAs without a proper feature extraction process. Such challenges require innovative approaches to the feature engineering and selection process. Pan et al. (2022), Wang and Guo (2021), Patsakis and Casino (2021) did not explicitly address the feature engineering process.

However, Kostopoulos et al. (2023) significantly improved the feature engineering process by leveraging 50 different features, resulting in an impressive 94% accuracy. In their experiments, Kostopoulos et al. (2023) utilised various features extracted from domain names to enrich their analysis, including ***Shannon entropy***,***Frequency s***, ***Vowel Frequency***, and ***domain length***. Each feature serves a distinct purpose, capturing different facets of domain names without extracting TLDs. For instance, ***Shannon entropy*** measures a random variable's average "surprisal", which is particularly applicable to continuous probability distributions (Zhao et al., 2023a; Brandstaetter, 2024). ***Frequency_s*** examines character distribution to identify anomalies, while ***Vowel Frequency and Length*** offer insights into the linguistic composition and domain name size, facilitating classification across different domain types. In contrast, the feature set employed by Kostopoulos et al. (2023) is comparatively narrower in scope than the one we proposed. Therefore, feature extraction is crucial in detecting DGAs effectively. Through detailed analysis and experimentation, it becomes evident that selecting and utilising appropriate features significantly impact the accuracy and efficacy of the DGA detection ML model. We introduce a Nature-inspired Optimised ML-based model (NIOM-DGA) to address these research gaps.

NIOM-DGA leverages the strengths of ML techniques, which offer adaptability and scalability compared to static blocklisting methods. NIOM-DGA is built on domain ***characteristics*** + ***algorithm*** features extraction; NIOM-DGA uses a large dataset that can effectively detect emerging threats posed by DGAs, which constantly evolve to evade detection. Building upon the previous ML model, NIOM-DGA also addresses gaps identified in earlier studies, such as adversarial techniques. NIOM-DGA employs a comprehensive set of diverse features and leverages nature-inspired optimisation for hyperparameter tuning to detect DGA-generated and benign domains effectively.

4. ExtraHop networks dataset description

Using a large dataset is crucial for training a model when detecting DGAs because of the sheer volume and diversity of domain names that can be generated by DGA malware. DGAs are designed to create many unique domain names, often hundreds of thousands or millions (Kemmerling, 2023; Patil et al., 2022). Malware frequently uses these domains to establish communication channels with command and control servers or to distribute malicious payloads. By training a model on a large dataset, the model can recognise the patterns and characteristics common to DGA-generated domain names. This method includes features such as the domain length, the presence of certain characters or character sequences, and statistical properties of the domain names. Using a large dataset ensures that the model is exposed to various DGA-generated domain names, making it more robust and capable of generalising to new, unseen examples (Sharma, 2023).

Consequently, in this research, we employed a recent and one of the most comprehensive dataset called the Extrahop Network dataset (Extrahop Network, 2024), which contains over 16 million balanced, benign and dga domains. Moreover, we employ another ten (10) datasets to evaluate and consolidate the performance of our model on external data sources. Table 2 summarises the training and testing description of the dataset. Our primary dataset originates from ExtraHop Networks, which was used to train NIOM-DGA. The ExtraHop Networks DGA detection dataset is publicly accessible on their GitHub repository. It is the cornerstone for our research or the only dataset used to train/test the model with an 80/20 split. Extrahop Network (2024). The ExtraHop Networks dataset is encapsulated within a single file, `dga-training-data-encoded-v3.json.gs`, containing an extensive collection of domain entries. Encoded in JSON format, each entry comprises of two fundamental components: the domain name and its corresponding threat classification. The classification distinguishes between benign domains and those generated by DGAs, providing invaluable insight into the prevalence of malicious activities.

This dataset is thoroughly curated, encompassing over 16 million domain entries. Furthermore, the distribution between benign and DGA-generated domains is approximately balanced, facilitating robust model training and evaluation. In preparation for analysis, we conducted comprehensive data cleaning procedures. Specifically, we removed extraneous metadata, such as the "threat" label, and transformed the dataset into a more accessible CSV format. Consequently, the cleaned CSV file presented a concise and structured dataset representation. Each entry in the dataset consists of a domain name paired with its corresponding label. We then transform each domain into binary feature representations that capture its algorithmic characteristics. These features are subsequently used to train and test the NIOM-DGA model.

4.1. External validation: NIOM-DGA training and testing datasets

While significant research efforts have been devoted to detecting DGAs (Highnam et al., 2021; Sivaguru et al., 2020; Suryotrisongko and Musashi, 2022; Tuan et al., 2022; Javed et al., 2023), there remains a need for further external validation experiments to assess the effectiveness of developed models. We recognise that external model validation is often overlooked despite its crucial role in ensuring the model's efficiency.

Table 2
Overview of all datasets used in conjunction with NIOM-DGA.

No.	Dataset (Ref)	Purpose	Samples	Description
1.	Extrahop Network Dataset (Extrahop Network, 2024)	Training/Testing	16,246,014	Contains 16 m balanced DGA and benign dataset.
2.	Aayush V Dataset (Shah, 2025)	Testing	16,773,525	Balance DGA and benign domain names.
3.	baderj Dataset (Bader, 2024)	Testing	265,538	Collection of DGAs generated domain names.
4.	DGArchive Dataset (DGArchive, 2025)	Testing	5632,234	Archive of DGAs generated domain names.
5.	harpomaxx Dataset (Harpo, 2023)	Testing	8342,022	Comprises both DGA and benign domain names.
6.	Abakumov Dataset (Abakumov, 2024)	Testing	825,991	Contained Benign and DGA generated domains names.
7.	Charles Dataset (Givre)	Testing	160,003	Dataset of DGA and benign domain names.
8.	siyad Dataset (Mestour, 2021)	Testing	1404,792	Comprises both DGA and benign domain names.
9.	OmurcanTATAR Dataset (Tatar, 2024)	Testing	1029,822	Contained Benign and DGA generated domains names.
10.	Sabin Dataset (Sharma Paudel)	Testing	1542,771	Umbrella benign domains and DGA domains names.
11.	Rafael Dataset (Gregório, 2023)	Testing	2017,746	Majestic legit benign and DGA domains and names.

Recognising this gap, we employed a comprehensive evaluation methodology by leveraging publicly available DGA datasets to evaluate the NIOM-DGA model thoroughly (Table 2). Understanding that the performance of machine learning classifiers is intrinsically tied to the quality and diversity of the training data (El-Ghamry et al., 2023,?; Mehta et al., 2023), we took a proactive approach by integrating character-based and algorithm-based detection techniques into NIOM-DGA. This combination enables NIOM-DGA to draw upon the strengths of each feature extraction process, thereby empowering it to make informed decisions in domain threat detection scenarios. It is important to note that the results of this extensive detection and evaluation endeavour will be thoroughly discussed in the experiment results Section, providing valuable insights into the efficacy and performance of the NIOM-DGA framework in real-world settings.

5. Nature-inspired optimised ML-based model (NIOM-DGA)

5.1. Research methodology

The study utilises two primary datasets as shown in Fig. 2: the Extrahop Network dataset and an External dataset. The Extrahop Network dataset provides real-world network traffic data, serving as a reliable basis for analysing the behaviour of Domain Generation Algorithms (DGAs). The External dataset is employed to validate the generalisability of the proposed model, ensuring its applicability across diverse scenarios. Both datasets undergo thorough preprocessing to prepare them for analysis. This involves cleaning the data to remove noise, normalising feature values for consistency, and addressing missing or incomplete entries. These preprocessing steps ensure that the datasets are optimised for the subsequent stages of feature extraction and analysis.

Feature extraction is carried out to derive meaningful attributes that effectively represent the behaviour of DGAs. This process focuses on identifying key patterns within the data, including character frequency distributions, Shannon entropy analysis, Jaro entropy values, and Vowel-Consonant ratio. These features are carefully designed to highlight the distinctive characteristics of malicious domains. To prioritise the most significant features, SHAP values (SHapley additive explanations) are employed as a robust interpretability technique as shown in Fig. 5. This approach quantifies the contribution of each feature, ensuring that only the most influential attributes are retained for training the machine learning models.

The machine learning models are trained using the Extrahop dataset, with the data divided into training and testing subsets to facilitate evaluation on unseen samples. During the training phase, experiments are conducted with various classifiers and their performance is assessed using key evaluation metrics. To further enhance the performance of the models, the hyperparameters are fine-tuned through nature-inspired optimisation techniques. This optimisation ensures that the models achieve peak performance on the training dataset, reducing the risk of overfitting while maximising predictive accuracy. The model, referred to as NIOM-DGA, is rigorously tested on both the Extrahop and External datasets. Its performance is evaluated using metrics such as accuracy, precision, recall, and F1 score, providing a comprehensive assessment of its effectiveness. The results are compared with the baseline methods to contextualise the advances achieved by the proposed approach.

An overview of the NIOM-DGA model is presented in Fig. 3. The proposed model integrates domain-specific features and employs advanced algorithms for character-based feature extraction compared to the previous DGA ML. The NIOM-DGA technique uses advanced feature extraction techniques to extract 78 distinct features from domain names to significantly enhance the discriminatory capabilities of classifiers, enabling more precise and reliable detection of Domain Generation Algorithm (DGA) attacks. In addition, we researched how these DGA are created, as discussed in the background Section, and we leveraged the same technique for feature extraction. Many previous works related to ML DGA detection in the literature are constrained by small or synthetic datasets, which limits their findings' generalisability and real-world applicability (Pan et al., 2022; Kostopoulos et al., 2023; Gogoi and Ahmed, 2023; Quezada et al., 2022).

The proposed NIOM-DGA research fills this gap by conducting comprehensive evaluations on large-scale, diverse datasets representative of real-world network traffic. The NIOM-DGA contains a sophisticated character-algorithm features extraction process. This extraction is an extensive set of 78 distinct features, as shown in Fig. 3 . These features form the basis for our machine learning (ML) models, which include Random Forest (RF), XGBoost (XGB), Decision Tree (DT), AdaBoost-Classifier, ExtraTreesClassifier, Multi-layer Perceptron (MLP), and Logistic Regression. To enhance the performance of the NIOM-DGA, we go a step further by optimising its parameters. We achieve this by leveraging nature-inspired algorithms such as the Bat Algorithm, Grey Wolf Optimise, and Firefly Algorithm (El-Ghamry et al., 2023,?; Mehta et al., 2023). These optimisation techniques fine-tune the best-performing ML algorithm's hyperparameters, thereby increasing our model's accuracy (Pye et al., 2020).

We rigorously test the NIOM-DGA to thoroughly evaluate its effectiveness. We employ 10 external datasets representing unique scenarios to assess how well our model detects unknown DGA-generated and benign domain names. This thorough evaluation process allows us to measure the NIOM-DGA's robustness and versatility across various real-world scenarios.

5.2. Data preprocessing and features extraction

Our data preprocessing ensured consistency and readiness for analysis. The dataset was already in a clean and structured JSON format, containing domain names and their classification labels (e.g., `{''domain'': ''wgegrlteegwrrrerwi'', ''threat'': ''dga''}`). We converted it to CSV for compatibility with our extraction tools, verified there were no duplicate entries, and focused on the key fields for feature extraction. We then applied additional feature extraction processes, as detailed in Table 3 including Domain_Length, Substring_Count,Special_Character_Presence, a-s_Frequency_with_Num-

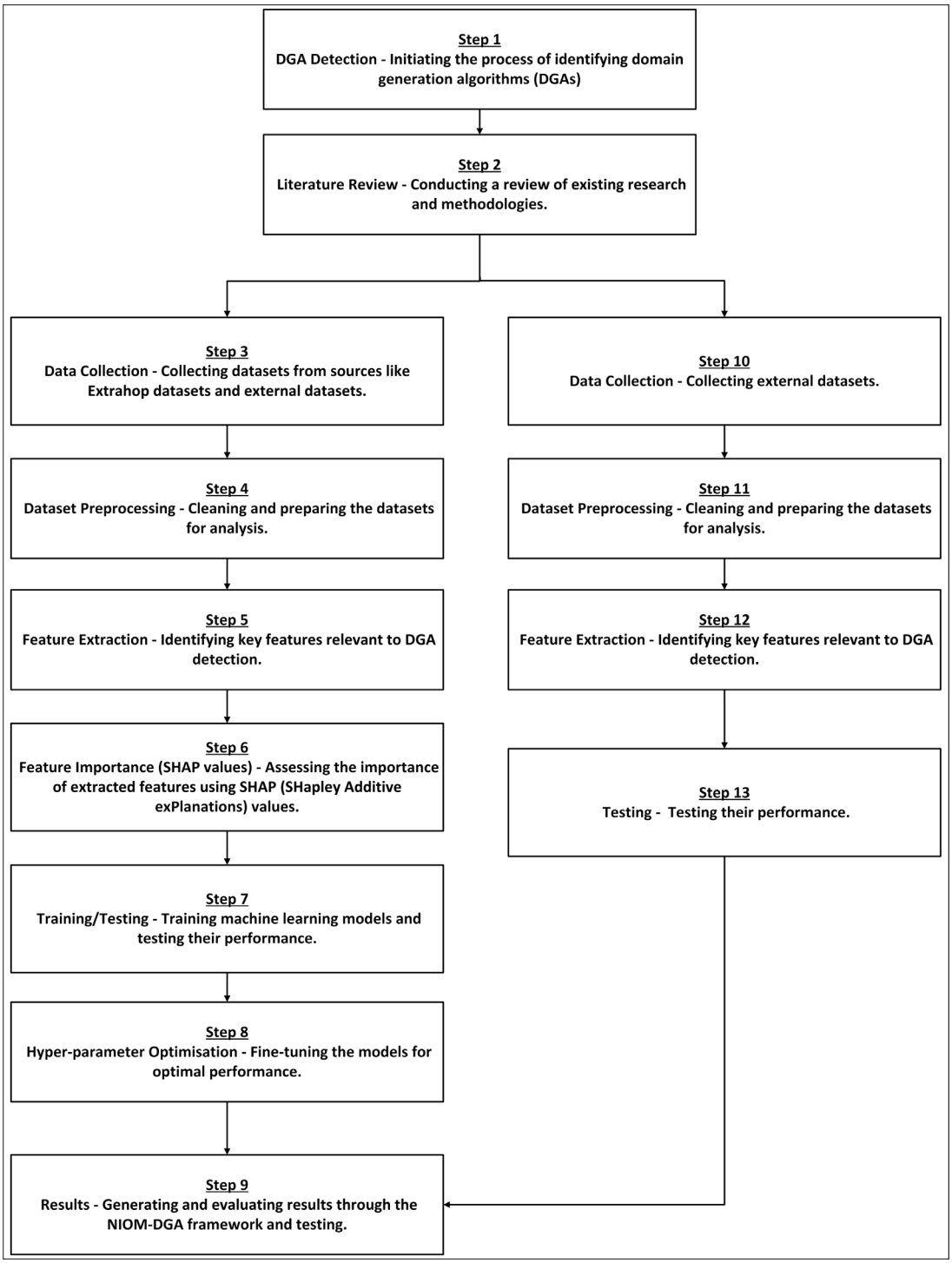


Fig. 2. Research design methodology.

bers and Substring_Count, preparing the data for effective machine learning training and testing. Additionally, applying advanced feature engineering techniques facilitated the extraction of pertinent features, enhancing the dataset's utility for comprehensive analysis and modelling.

Table 3 presents an overview of features which are extracted from domain names in this study. While most of the feature type names shown in Table 3 are self-explanatory, in this section, we explain ***Shannon_Entropy, Jaro_Similarities, Cosine_Similarity***, ***Needleman_Wunsch_Feature, and Smith_Waterman_Feature***.

Shannon_Entropy: Shannon Entropy serves as a metric to measure the randomness or unpredictability of a domain name. Originally proposed by Claude Shannon in information theory, this measure quantifies the uncertainty or disorder within a given data set (Kostopoulos et al., 2023). In the context of domain names, Shannon Entropy helps discern patterns or irregularities that may indicate suspicious or algorithmically generated domains. Higher entropy values suggest greater complexity and randomness, potentially signalling the presence of a DGA (Hsu et al., 2021). The Shannon Entropy $H(X)$ of a discrete random variable X with probability mass function $P(X)$ is calculated

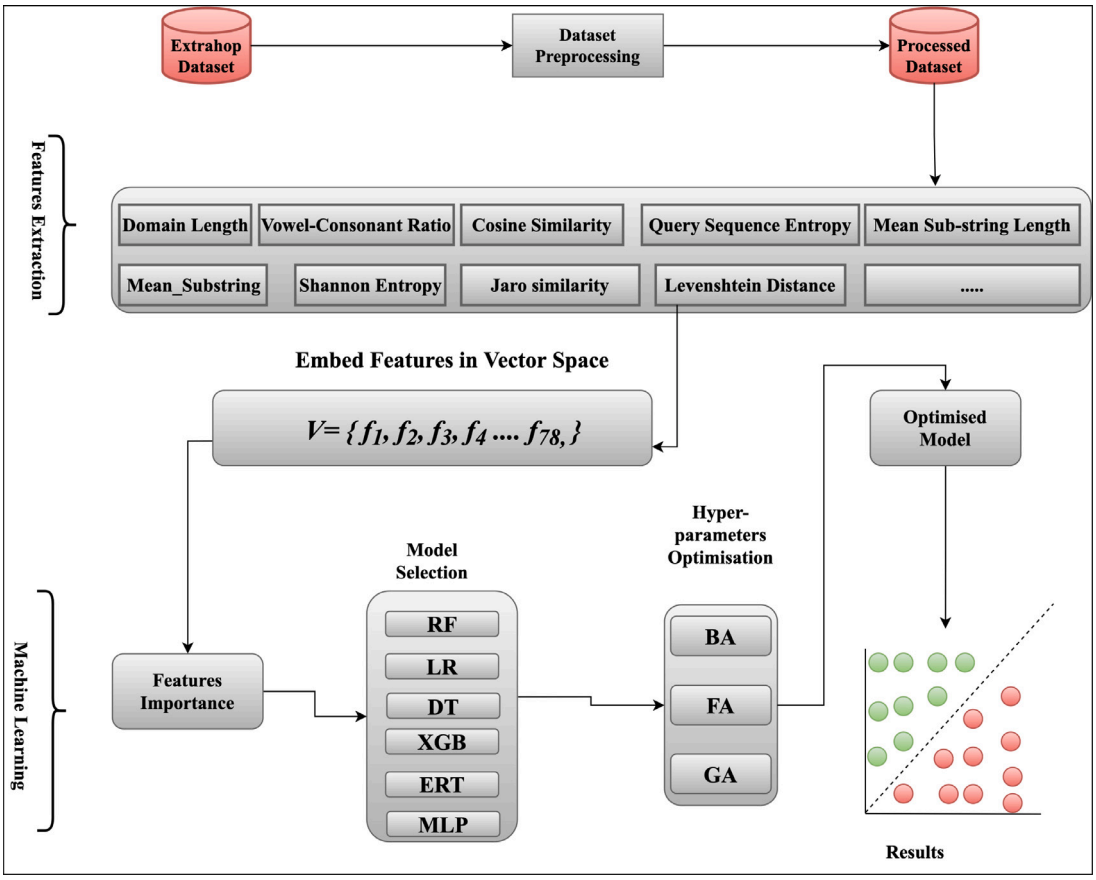


Fig. 3. Block Diagram of NIOM-DGA.

Table 3
Feature name(s), Sequence number, and Descriptions.

Feature name(s)	Sequence number	Description
Domain_Length	1	Length of the domain name
Shannon_Entropy	2	Measure of randomness or unpredictability of a domain name
Jaro_Similarities	3	Measure of similarity between two strings
Cosine_Similarity	4	Measure of cosine angle between two vectors
Needleman_Wunsch_Feature	5	Algorithm for global sequence alignment
Smith_Waterman_Feature	6	Algorithm for local sequence alignment
Character_Set_Diversity	7	Number of unique characters in the domain
Levenshtein_Distance	8	Minimum number of single-character edits
Query_Sequence_Entropy	9	Measure of uncertainty or randomness of a query sequence
Vowel_Cluster_Count	10	Count of consecutive vowel sequences
A_s_with_Numbers_Character	11–36	Presence of specific characters (A-s) with numbers
Consonant_Cluster_Count	37	Count of consecutive consonant sequences
D_Character_Count	38	Count of the letter 'D'
Dash_Presence	39	Presence of a dash character
Date_Time_Character_Presence	40	Presence of date and time characters
Double_Character_Presence	41	Presence of repeated characters
Consecutive_Sequence_Length	42	Length of the longest consecutive character sequence
Lowercase_Letter_Count	43	Count of lowercase letters
Mean_Substring_Length	44	Mean length of all substrings
Numeral_Presence	45	Presence of numeral characters
Special_Character_Presence	46	Presence of special characters
Substring_Count	47	Count of all substrings
Unconventional_Structure_Presence	48	Presence of unconventional domain structures
Unique_Character_Count	49	Count of unique characters
Unique_Character_Ratio	50	Ratio of unique characters to total characters
Unique_Substring_Count	51	Count of unique substrings
Vowel_Consonant_Ratio	52	Ratio of vowels to consonants
a-s_Frequency_with_Numbers	53–78	Frequency of lowercase letters (a-s)

as:

$$H(X) = -\sum_{i} P(X = x_i)\log_2(P(X = x_i)) \quad (1)$$

Where x_i represents each possible outcome of the random variable X.

Jaro_Similarities:

Jaro Similarity is a metric used to determine the similarity between two strings, primarily focusing on character matches and transpositions (Basak et al., 2023). It calculates the proportion of matching characters between two strings, such as benign and dga domains, considering the number of matching characters and their positions. By quantifying the degree of resemblance between domain names, Jaro Similarity aids in distinguishing legitimate domains from those generated by DGA. The Jaro similarity coefficient **J** between two strings $s1$ and $s2$ of lengths len_s1 and len_s2 respectively is defined as:

$$J = \frac{1}{3}\left(\frac{\mathbf{m}}{len\_s1} + \frac{\mathbf{m}}{len\_s2} + \frac{\mathbf{m} - t}{\mathbf{m}}\right) \quad (2)$$

Here, **m** represents the count of matching characters, while t denotes the number of transpositions—instances where matching characters are not in the correct sequence (Sahni and Rajasekaran, 2023).

Jaro Similarity provides a numerical indication of how closely two strings resemble each other, with $J = 1$ indicating identical strings and $J = 0$ implying no character matches.

Cosine_Similarity: Cosine Similarity measures the cosine of the angle between two vectors in a multi-dimensional space, commonly used in natural language processing and text mining tasks (Kirişci, 2023; Hasan and Ferdous, 2024). We leverage cosine Similarity to assess the similarity between domain name feature vectors extracted from datasets. By evaluating the geometric relationship between feature vectors, Cosine Similarity helps identify clusters or patterns indicative of DGA-generated domains. Higher cosine similarity values imply greater resemblance between domain features, assisting in classifying malicious and benign domains (Kirişci, 2023; Hasan and Ferdous, 2024). The Cosine Similarity cosine_similarity between two feature vectors **A** and **B** is calculated as:

$$\text{cosine\_similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \quad (3)$$

Where $\mathbf{A} \cdot \mathbf{B}$ represents the dot product of the two vectors, and $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ are the magnitudes of the vectors.

Needleman_Wunsch_Feature: The Needleman–Wunsch feature leverages this algorithm to align pairs of domain names, identifying conserved regions and gaps between characters between two strings (Hu et al., 2024; Likić, 2007). In this case its between benign domains and dga domains. By aligning sequences and assigning similarity scores based on matches, mismatches, and gaps, the Needleman–Wunsch feature aids in quantifying the degree of resemblance or divergence between these domain names. This facilitates the detection of subtle variations and patterns characteristic of DGA-generated domains. The Needleman–Wunsch alignment score $\text{NW}(s1, s2)$ between two strings $s1$ and $s2$ is calculated using dynamic programming as:

$$\text{NW}(s1, s2) = \max_{i,j}\{M_{i,j}\} \quad (4)$$

Where $M_{i,j}$ represents the score of the best alignment ending at position i in string $s1$ and position j in string $s2$.

Smith_Waterman_Feature: Similar to the Needleman–Wunsch algorithm, the Smith–Waterman algorithm is also employed for sequence alignment but focuses on local sequence similarity rather than global alignment (Chagneau et al., 2024; Mehri et al., 2023). The Smith–Waterman feature evaluates pairs of benign and dga domain names to identify regions of maximal similarity within a regional context. By pinpointing areas of significant overlap or divergence between domain sequences, the Smith–Waterman feature enhances the granularity of similarity assessment, offering insights into specific motifs or patterns associated with DGA-generated domains. The Smith–Waterman alignment score $\text{SW}(s1, s2)$ between two strings $s1$ and $s2$ is calculated similarly to Needleman–Wunsch, but allowing negative scores and selecting the maximum local score instead of the global score.

$$\text{SW}(s1, s2) = \max_{i,j}\{M_{i,j}\} \quad (5)$$

Where $M_{i,j}$ represents the score of the best alignment ending at position i in string $s1$ and position j in string $s2$.

As depicted in Fig. 4, showcasing SHAP feature importance grouped into eight categories, we evaluated the model to ensure the significance of all features contributing to the results. Initially, our feature set comprised over 100 variables. However, as detailed in Section 5.2, certain features proved impractical and were consequently eliminated to enhance model efficiency using SHAP as shown in Fig. 5. While these removals optimised the model, we attempted to remove the '**special_character_presence**' feature due to its limited use, as indicated by SHAP analysis. Unfortunately, this adjustment resulted in a marginal decrease in model accuracy by 0.4%.

5.3. NIOM-DGA features importance

Feature importance is crucial during the ML training phase; these features encapsulate the essential characteristics of the data and play a pivotal role in determining the performance and effectiveness of ML models. Properly engineered features enable algorithms to identify patterns, make accurate predictions, and derive meaningful insights from complex datasets. In addition, the quality and relevance of ML features directly impact the success and efficacy of machine learning applications across various domains, making them a critical component of the ML workflow.

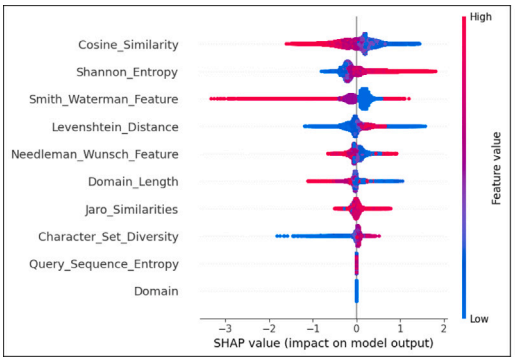
Subsequently, Effective feature selection and engineering are pivotal as they directly impact the model's ability to learn meaningful patterns and make accurate predictions. During our research, we identified over 100 features. However, upon assessing feature importance, we found that some features could have contributed more effectively to the machine learning process, so we removed them. These included ***Non-ASCII_Character_Presence***, ***Misspelled_Words_Presence***, ***IDN_Homograph_Presence***, and ***Cyrillic_or_Chinese_Presence***.

To understand feature importance, we utilised XGBoost alongside SHAP (SHapley Additive exPlanations), a method in ML that explains individual predictions by assigning values to features (Kostopoulos et al., 2023). SHAP quantitatively assesses the contribution of each feature to model predictions using the formula:

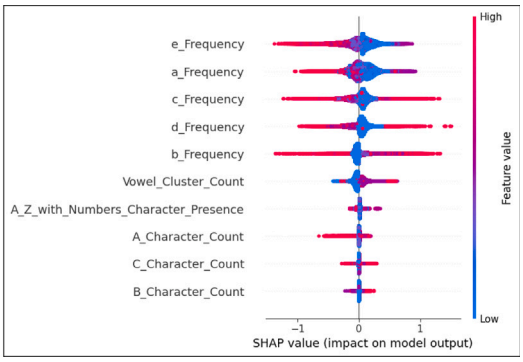
$$\boldsymbol{\phi}_i(f) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!}[f(S \cup \{i\}) - f(S)] \quad (6)$$

The Shapley value $\boldsymbol{\phi}_i(f)$ for feature i in function f. It considers all possible subsets S of features excluding i, calculates the difference in model output when adding feature i to subset S, and averages these differences over all subsets. This provides a fair assessment of each feature's contribution to the model's predictions.

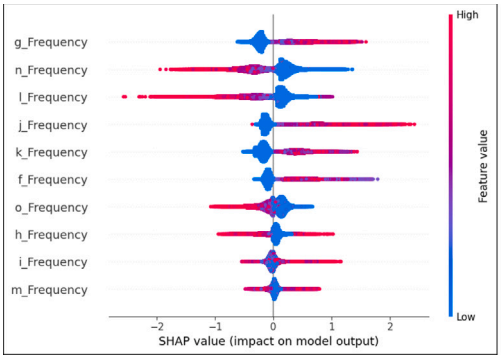
In Fig. 5, we present an analysis of features derived from SHAP (SHapley Additive exPlanations) values, which quantitatively assess the contribution of each feature to the model's predictions. In Fig. 5(a) to (f), we examine the interconnected narratives of various feature pairs shaping model predictions. In these plots, features that dominantly influence name classifications are depicted along with their values. Red denotes features related to DGA domain names, and blue colours contribute to benign domain names. Consequently, the interaction between ***Character_Set_Diversity*** and ***Cosine_Similarity*** reveals modest variations in ***Character_Set_Diversity***, while ***Cosine_Similarity*** exhibits a more pronounced influence. Another feature we analyse is ***Shannon_Entropy's*** diverse impact, which contrasts with ***Cosine_Similarity's*** less pronounced influence, shaping the model's understanding of the data. In addition, we analyse the ***Levenshtein_Distance***, while the feature impacts predictions moderately, while ***Shannon_Entropy's*** broader impact contributes to the model's prediction. Furthermore, ***Cosine_Similarity*** and ***Mean_Substring_Length*** interact moderately,



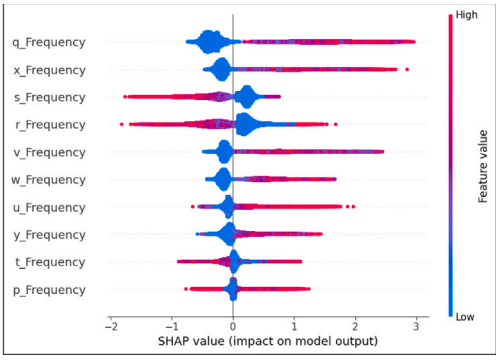
(a) Group 1: Shap importance values for top contributing features



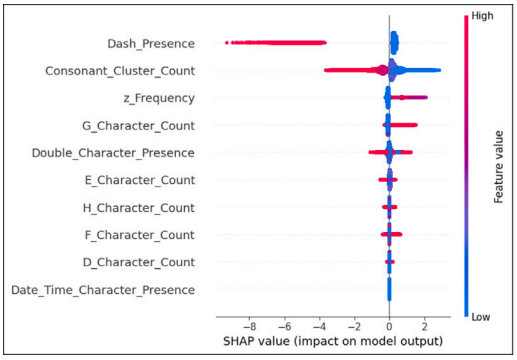
(b) Group 2: Shap importance values for top contributing features



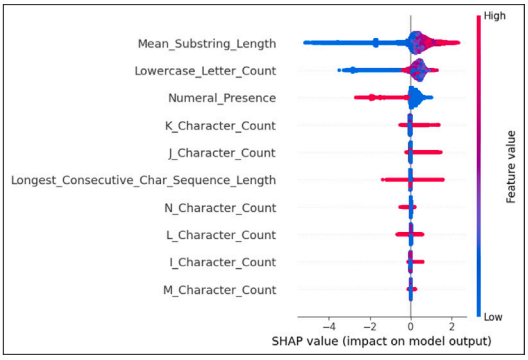
(c) Group 3: Shap importance values for top contributing features



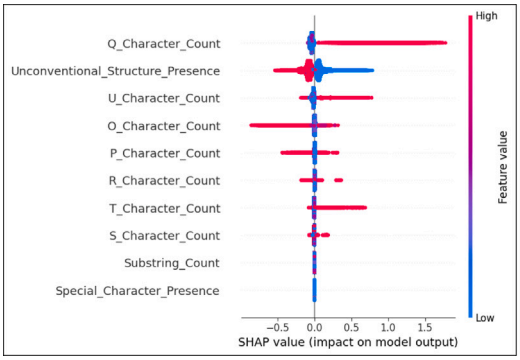
(d) Group 4: Shap importance values for top contributing features



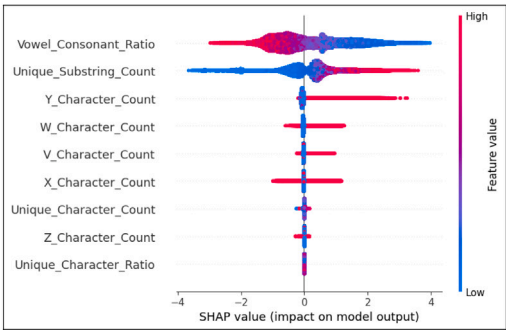
(e) Group 5: Shap importance values for top contributing features



(f) Group 6: Shap importance values for top contributing features

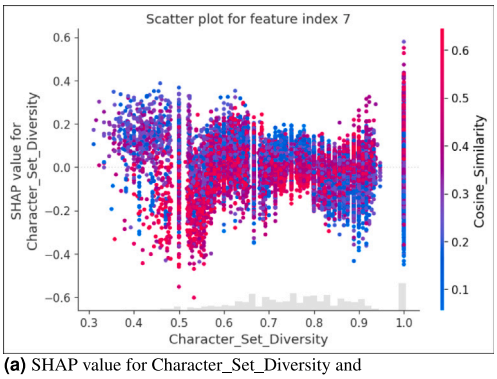


(g) Group 7: Shap importance values for top contributing features

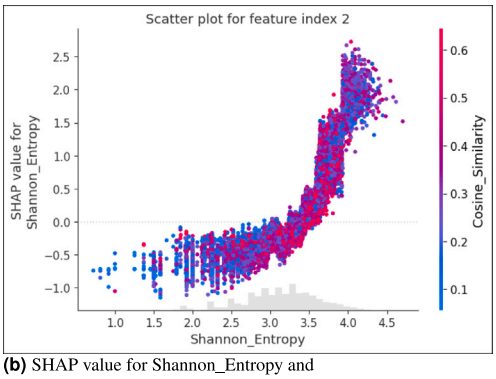


(h) Group 8: Shap importance values for top contributing features

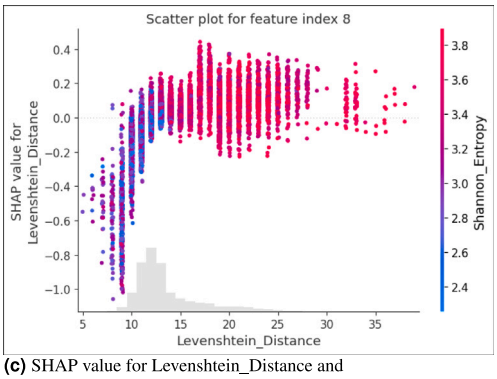
Fig. 4. SHAP feature importance grouped into eight categories.



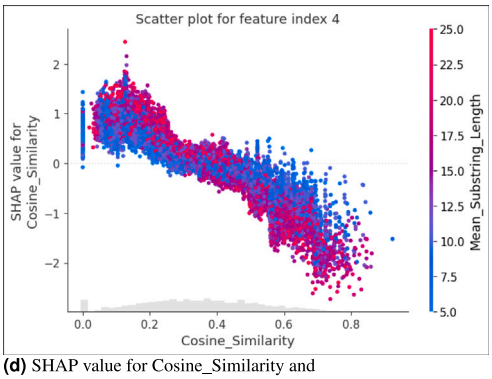
(a) SHAP value for Character_Set_Diversity and Cosine_Similarity



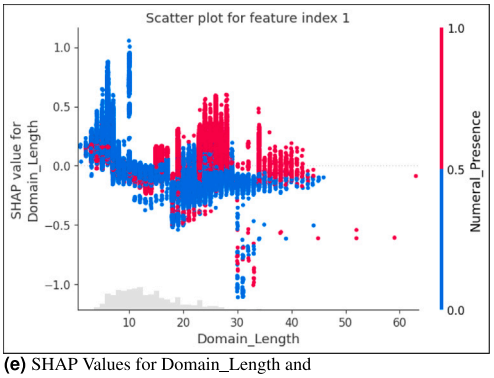
(b) SHAP value for Shannon_Entropy and Cosine_Similarity



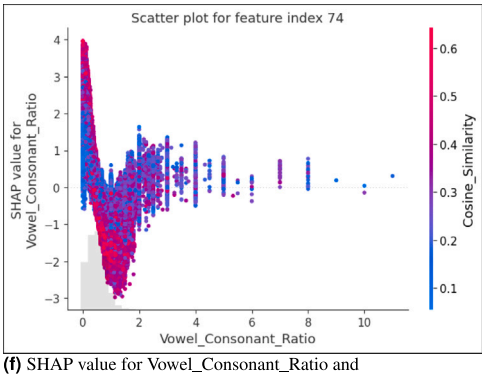
(c) SHAP value for Levenshtein_Distance and Shannon_Entropy



(d) SHAP value for Cosine_Similarity and Mean_Substring_Length



(e) SHAP Values for Domain_Length and Numeral_Presence



(f) SHAP value for Vowel_Consonant_Ratio and Cosine_Similarity

Fig. 5. Empirical analysis of SHAP values for feature pair interactions: red indicates DGA traffic; blue indicates benign traffic. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

showcasing a more comprehensive influence range alongside the ***Cosine_Similarity***. Consequently, we also look at the ***Domain_Length*** and ***Numeral_Presence***, which complement the model's effectiveness. Lastly, we analyse that ***Vowel_Consonant_Ratio*** significantly influences predictions, while ***Cosine_Similarity's*** effect is less pronounced. The SHAP scatter plots unveil better feature effectiveness interactions shaping model predictions.

5.4. Learning and experimental phase

NIOM-DGA employs a variety of machine learning algorithms, including Random Forest (RF), Logistic Regression (LR), Decision Trees (DT), XGBoost (XGB), AdaBoost (AB), Extremely Randomised Trees (ERT), and Multilayer Perceptron (MLP) for model training using the ExtraHop Dataset with 80/20 train test splits. Following classification, the top-performing model is selected and subjected to further refinement using nature-inspired optimisation algorithms. These algorithms include the bat algorithm (BA), grey wolf optimiser (GWO) and firefly algorithm (FA). The aim is to enhance classification accuracy by fine-tuning the selected model.

We evaluate the classification results of ML algorithms based on the outcomes of the confusion matrix. Confusion matrix summaries the results of machine learning classifiers based on correct and incorrect predictions by using the following metrics:

- **True Positive (TP):** The number of DGA domains correctly classified as DGA.
- **True Negative (TN):** The number of benign domains correctly classified as benign.
- **False Positive (FP):** The number of benign domains incorrectly classified as DGA.
- **False Negative (FN):** The number of DGA domains incorrectly classified as benign.

The performance metrics we consider include accuracy (Eq. (7)), recall (Eq. (8)), precision (Eq. (9)), and F1-score (Eq. (10)), which are

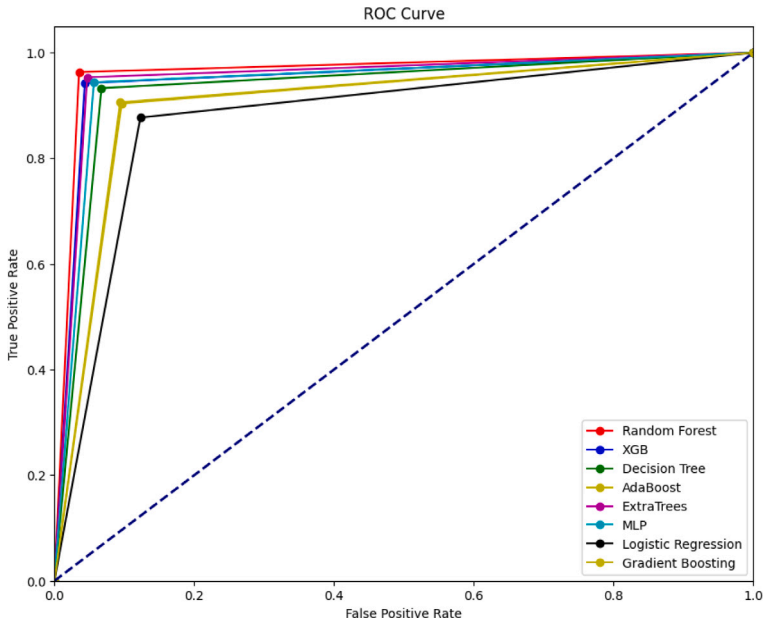


Fig. 6. ROC Curves of Classifiers Trained with ExtraHop Networks Dataset.

Table 4
Performance comparison of ML classifiers.

Classifier	Accuracy	Recall	Precision	F-measure
RF	96	95	96	96
XGB	95	94	95	94
DT	93	93	93	93
AdaBoost	91	90	90	90
ExtraTrees	96	95	96	95
MLP	94	94	94	94
LR	90	87	92	90
GB	92	90	93	91

derived from the confusion matrix.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

Table 4 presents the classification results of NIOM-DGA by employing the ExtraHop Network dataset. As shown in Table 4, RF outperforms all the other classifiers (ExtraTrees, XGB, DT, Adaboost, MLP, LR and GB) in terms of classification results. Similarly, Fig. 6 illustrates the receiver operating characteristic (ROC) curves obtained from classifiers trained on the Extrahop Network dataset.

As shown in Fig. 6, the ROC curves depict the false positive rate (FPR) on the x-axis and the true positive rate (Recall) on the y-axis. These curves reveal significant outcomes, with Random Forest as compared to ExtraTrees, XGB, DT, Adaboost, MLP, LR and GB. Subsequently, to further increase the efficacy of NIOM-DGA, we integrate NIAs to determine the optimal hyper-parameter settings for the most effective classifier (RF) (Rafiq et al., 2022). Our considerations contain the Bat algorithm (BA), Firefly algorithm (FA), and Grey Wolf optimiser (GWO) for tuning the hyper-parameters of RF.

5.5. Random forest hyper-parameters proposed by NIAs

We employ NIAs for hyper-parameter optimisation, leveraging their ability to navigate complex optimisation spaces effectively (Rafiq et al., 2022). Unlike traditional methods, NIAs draw inspiration from natural phenomena, mimicking processes like evolution and swarm behaviour to tackle challenging optimisation problems (Mehta et al., 2023; El-Ghamry et al., 2023). They have a proven track record of success in various applications, particularly in optimising the hyperparameters of complex machine-learning models (Mehta et al., 2023; Pye et al., 2020). By mimicking natural processes, NIAs offer a unique approach to exploring and exploiting the search space, often leading to superior solutions. They exhibit robustness, scalability, and versatility, making them suitable for optimising diverse machine-learning algorithms. NIAs handle high-dimensional parameter spaces and non-linear relationships efficiently, empowering effective fine-tuning of models to enhance performance and generalisation ability.

Table 5
Hyper-parameters for RF proposed by NIAs.

Hyper-parameter	BA	FA	GWO
n_estimators	60	80	80
max_depth	34	28	28
min_sample_split	2	2	2
max_features	*auto*	*sqrt*	*auto*

During the training phase, we evaluated multiple machine learning algorithms, including Random Forest (RF), XGBoost (XGB), Decision Trees (DT), AdaBoostClassifier, Extra Trees Classifier, MLP Classifier, Logistic Regression, and Gradient Boosting Classifier. Each algorithm underwent rigorous scrutiny, focusing on pivotal performance metrics such as accuracy, recall, precision, and F-measure. Among these algorithms, Random Forest (RF) outperformed all the other classifiers, showcasing remarkable accuracy at 96%. In order to achieve even better results, we employ NIAs to tune the hyper-parameters of RF classifiers for DGA detection.

Table 5 present the optimal hyper-parameters setting for RF determined by NIAs for classifying DGA-generated and benign domain names. Subsequently, Table 6 presents the classification results achieved by NIOM-DGA, a DGA-generated and benign domains classifier based on RF and optimised using NIAs. Compared to the RF classifier results in Table 4 and Fig. 7, NIOM-DGA remarkably strengthens the performance by employing NIAs to determine the optimal setting of hyper-parameters (upto 98% accuracy in case of BA) (see Tables 7–11).

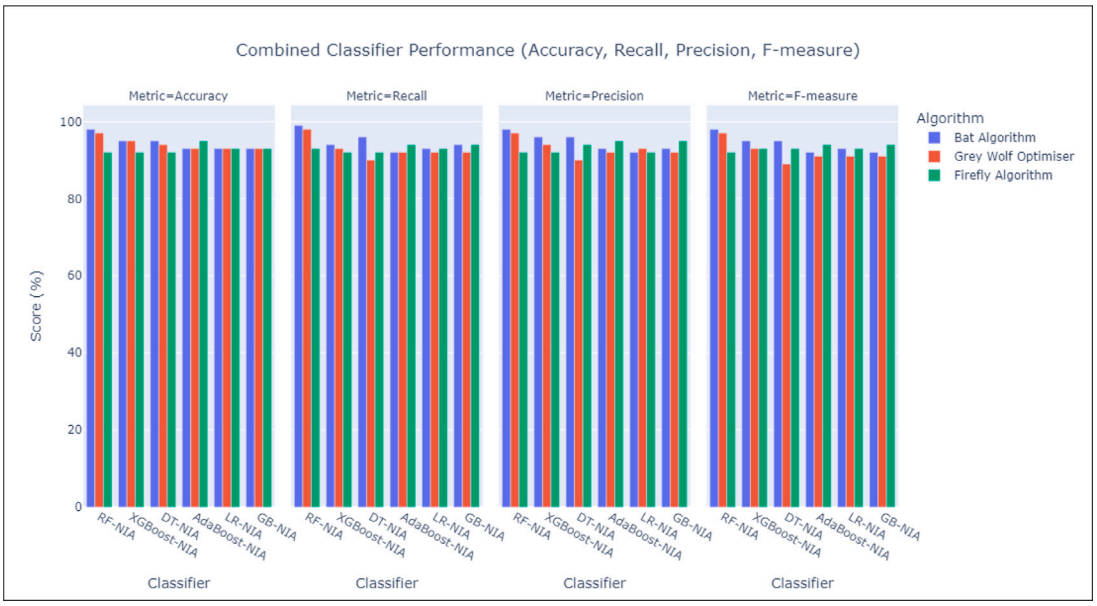


Fig. 7. Combined Classifier Performance after NIA Optimisation (Accuracy, Recall, Precision, F-measure).

Table 6
Performance of Random Forest after NIA optimisation.

Classifier	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)
Bat Algorithm	98	99	98	98
Grey Wolf Optimiser	97	98	97	97
Firefly Algorithm	92	93	92	92

Table 7
Performance of XGBoost after NIA optimisation.

Classifier	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)
Bat Algorithm	95	94	96	95
Grey Wolf Optimiser	95	93	94	93
Firefly Algorithm	92	92	92	93

Table 8
Performance of Decision Tree after NIA optimisation.

Classifier	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)
Bat Algorithm	95	96	96	95
Grey Wolf Optimiser	94	90	90	89
Firefly Algorithm	92	92	94	93

Table 9
Performance of AdaBoost after NIA optimisation.

Classifier	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)
Bat Algorithm	93	92	93	92
Grey Wolf Optimiser	93	92	92	91
Firefly Algorithm	95	94	95	94

Table 10
Performance of Logistic Regression after NIA optimisation.

Classifier	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)
Bat Algorithm	93	93	92	93
Grey Wolf Optimiser	93	92	93	91
Firefly Algorithm	93	93	92	93

Many model classifiers performed better with nature-inspired optimisation (NIA) techniques, which can be attributed to a combination of hyperparameter settings and the inherent characteristics of each optimisation algorithm.

Table 11
Performance of Gradient Boosting after NIA optimisation.

Classifier	Accuracy (%)	Recall (%)	Precision (%)	F-measure (%)
Bat Algorithm	93	94	93	92
Grey Wolf Optimiser	93	92	92	91
Firefly Algorithm	93	94	95	94

From the data in the tables, it is evident that the ***BA*** consistently outperforms the other nature-inspired optimisation techniques, such as the ***GWO*** and the ***FA***, across various classifiers. This superior performance can be attributed to the specific hyperparameter settings chosen for each model, which are tailored for the optimal functioning of BA. For instance, BA used a higher number of ***n_estimators*** (60 for BA, compared to 80 for GWO and FA), which may contribute to more robust learning through increased model complexity and diversity. Additionally, BA's use of ***max_depth*** values that are slightly higher (34 for BA) might help capture more complex patterns in the data, leading to better generalisation and higher performance metrics such as accuracy, recall, and precision.

In contrast, the ***GWO*** shows strong performance, particularly in Random Forest and XGBoost classifiers, where it performs second to BA. The ***max_depth*** of 28, a lower value than BA, suggests that GWO might be better suited for simpler models, avoiding overfitting while still achieving high performance. GWO's lower number of ***n_estimators*** compared to BA may be compensated by its efficiency in optimising hyperparameters, achieving solid performance with fewer estimators.

The ***FA***, despite its more conservative hyperparameter settings (with ***n_estimators*** at 80 for both FA and GWO), consistently trails behind the other two in terms of overall performance. This suggests that FA may require further tuning of its parameters or may not be as effective in capturing complex patterns in the dataset compared to BA and GWO.

Thus, the differences in performance are influenced by the interplay between the hyperparameters and the optimisation strengths of each algorithm. The Bat Algorithm, with its ability to explore the hyperparameter space more effectively, results in more optimal configurations, leading to better classification performance. The ***GWO***, with a balanced approach to hyperparameter selection, also performs well but not to the same extent as BA. The ***FA***, while effective in certain contexts, may need further refinement to compete with BA and GWO in terms of classifier performance.

Table 12
NIOM-DGA: Performance on external datasets.

No.	Dataset (Ref)	Samples	Accuracy	Precision	Recall	F-measure
1.	Aayush V Dataset (Shah, 2025)	16,773,525	0.890	0.893	0.880	0.718
2.	baderj Dataset (Bader, 2024)	265,538	0.968	0.980	0.968	0.959
3.	DGArchive Dataset (DGArchive, 2025)	5632,234	0.991	0.991	0.991	0.991
4.	harpomaxx Dataset (Harpo, 2023)	8342,022	0.981	0.911	0.971	0.981
5.	Abakumov Dataset (Abakumov, 2024)	825,991	0.971	0.991	0.961	0.991
6.	Charles Dataset (Givre)	160,003	0.991	0.981	0.991	0.961
7.	siyad Dataset (Mestour, 2021)	1404,792	0.896	0.893	0.878	0.885
8.	OmurcanTATAR Dataset (Tatar, 2024)	1029,822	0.931	0.915	0.925	0.920
9.	Sabin Dataset (Sharma Paudel)	1542,771	0.965	0.961	0.895	0.893
10.	Rafael Dataset (Gregório, 2023)	2017,746	0.906	0.895	0.905	0.900

Table 13
Comparison with existing works with NIOM-DGA.

Proposed model	Year	External experiment	ML algorithms	Features number	Testing amount	Accuracy
DGA-RF (Hoang and Vu, 2021)	2021	X	Random Forest	42	1500,000	90%
AGD - FANCI (Wang and Guo, 2021)	2021	X	LSTM	21	1500,000	78%
AGD - LSTM (Wang and Guo, 2021)	2021	X	Bi-LSTM	21	300,000	86%
AGD - Bi-LSTM (Wang and Guo, 2021)	2021	X	CNN	21	190,000	88%
AGD - CNN (Wang and Guo, 2021)	2021	X	CNN-LSTM	21	180,000	89%
AGD-CNN-LSTM (Wang and Guo, 2021)	2021	X	LSTM-Att	21	210,000	89%
Bi-LSTM (Pan et al., 2022)	2022	✓	LSTM	12	220,000	97%
HAGDetector (Liang et al., 2022)	2022	✓	LR	21	300,000	95%
CANINE -C (Gogoi and Ahmed, 2023)	2023	X	CANINE	21	90,000	99%
BiGRU-ATT (Yan et al., 2023)	2023	✓	BiGRU	20	1000,000	96%
MLP- DGA (Abhiram et al., 2023)	2023	X	MLP	7	700,000	97.24%
DGA-SVM (Moşolea and Oprişa, 2023)	2023	✓	SVM	30	600,000	96%
SESAME (Weissgerber et al., 2023)	2023	✓	SESAME	28	900,000	83.89%
DGA-T-C (Ding et al., 2023)	2023	✓	CNN	18	1000,000	96%
XAI x SHAP (Kostopoulos et al., 2023)	2023	X	XGBoost	50	900,000	94.81%
Bi-LSTM (Hassaoui et al., 2024)	2024	X	LSTM	21	1000,000	95%
HDDN (Chen et al., 2024)	2025	X	CNN + LSTM	39-char + 75 seq	674,898	93.86%
HDDN (Chen et al., 2024)	2025	X	CNN + LSTM	39-char + 75 seq	1000,000	90.09%
NIOM-DGA	**2025**	✓	**Random Forest**	**78**	**3200,000**	**98.3%**

After optimising the model with NIAs, we selected the best NIAs classifiers. We conducted further evaluations employing 10 external datasets (Table 2) for further testing. We considered the hyper-parameters setting determined by BA as it demonstrated superior performance compared to FA and GWO hyper-parameters settings when evaluated using the base dataset (Table 6). The combined samples in the external datasets for testing cover over thirty-seven million (37,883,844 approx.) benign and DGA-generated domain names, showcasing the comprehensive scope of the external evaluation conducted with NIOM-DGA. Table 12 presents the performance of NIOM-DGA on external datasets. As shown in Table 12, NIOM-DGA performs remarkably well when tested on the external datasets with an average accuracy of 95.7%.

Alongside classification outcomes, we evaluate the time complexity of NIAs as a performance metric for NIOM-DGA. Fig. 8 illustrates the time each NIA (BA, FA, and GWO) takes to optimise the hyper-parameters of RF across ten distinct external datasets. Each NIA began with a population size of 50, with a maximum of 100 iterations. BA demonstrates superior performance over FA and GWO in terms of time complexity and classification outcomes. FA and GWO also deliver commendable classification results; however, they require significantly more time than BA to identify optimal hyper-parameters across each dataset. Consequently, NIOM-DGA favours BA over FA and GWO for hyper-parameter optimisation to enhance RF performance in DGA classification.

6. Comparative analysis with other approaches

Table 13 compares NIOM-DGA with existing techniques proposed in the literature. As shown in the Table 13, only few of the related works (Bi-LSTM (Hassaoui et al., 2024; Pan et al., 2022), BiGRU-ATT (Yan et al., 2023), SVM (Moşolea and Oprişa, 2023), SESAME (Weissgerber et al., 2023), and T-C (Ding et al., 2023)) conducted external experiments to evaluate their models. In contrast, although NIOM-DGA achieved up to 98% classification accuracy by using the base dataset (ExtraHop Network dataset), it also considered ten external datasets for validation. It achieved up to 95.7% classification accuracy on average. The number and quality of features directly impact a model's ability to capture intricate patterns within the data. As shown in Table 13, NIOM-DGA employs up to 78 distinct features extracted from domain names, significantly more than those listed in the comparison table. However, models with higher feature numbers often possess a greater capacity for distinguishing complex relationships at the risk of over-fitting. Conversely, models with fewer features might exhibit improved generalisation but overlook crucial variations in the dataset. NIOM-DGA addresses these issues by thoroughly selecting the essential features for the model, with 78 distinct features. Finally, in terms of accuracy, apart from CANINE-C (Gogoi and Ahmed, 2023), NIOM-DGA significantly outperforms all the reported techniques in Table 13. Although CANINE-C (Gogoi and Ahmed, 2023) reports up to 99% classification accuracy, their testing set size is limited (up to 90,000). In contrast, NIOM-DGA employs over 3.2 Million samples from the ExtraHop Network dataset for testing and reports up to 98% classification accuracy.

7. Conclusion and future work

Our investigation into Domain Generation Algorithm (DGA) detection commenced with a thorough review of existing methodologies and datasets. To establish a robust foundation for the study, we drew upon a range of publicly available datasets and prior research (Chen et al., 2024; Kostopoulos et al., 2023; Abakumov, 2024; Javed et al., 2023; Pye et al., 2020). To address the inherent challenges associated with DGA detection, we employed the recent and most comprehensive ExtraHop dataset (to the best of our knowledge) comprising over 16 million DGA and benign domain names. This dataset consists of over 53 different DGA families, ensuring an extensive representation of

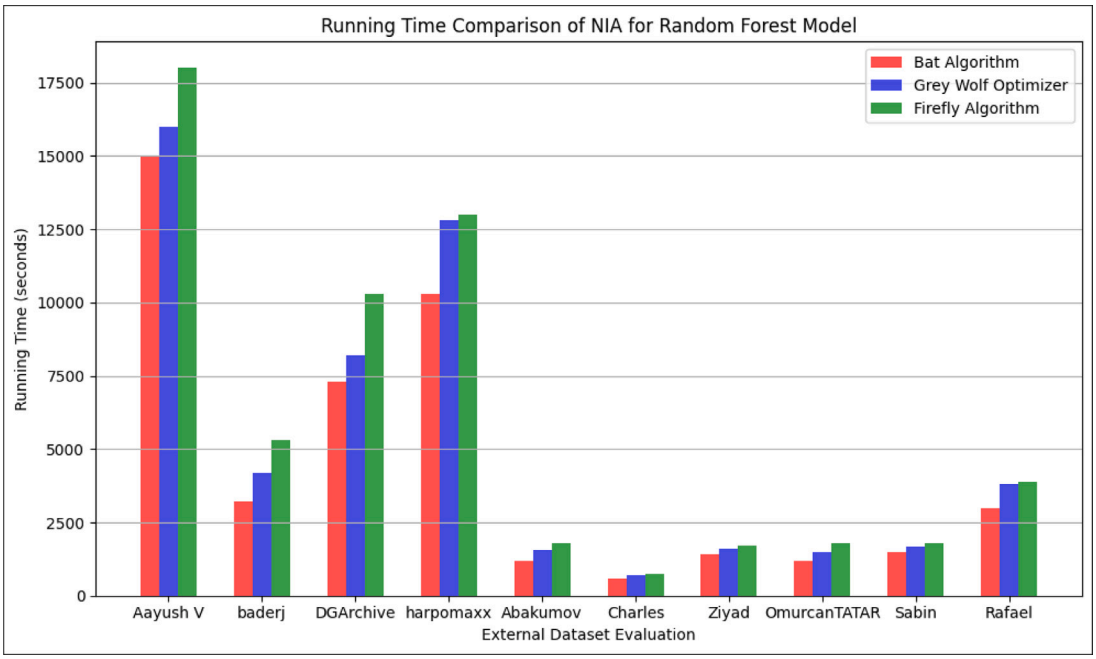


Fig. 8. Running Time Comparison of NIA for Random Forest Model.

the various techniques utilised by malicious actors to generate DGAs. Drawing upon insights from previous research, we thoroughly curated this dataset to ensure its representatives of real-world dga domain patterns. We used advanced feature extraction techniques to extract and integrate 78 distinct features from the domain names, ranging from linguistic properties to structural complexity and algorithm-based feature extraction. This holistic approach allowed us to significantly enhance the discriminatory capabilities of our classifiers significantly, facilitating more precise and reliable detection of DGA attacks.

Furthermore, we propose the NIOM-DGA ML model trained on 78 characteristic-algorithmic features extracted from the Extrahop Network Dataset and then optimised using nature-inspired algorithms. Our experiment results show that NIOM-DGA achieves up to 98% classification accuracy on the Extrahop Network dataset. To further consolidate the performance of our proposed model, we tested NIOM-DGA by employing 10 external datasets consisting of over 37 million domain names and achieved over 95.7% accuracy on average. Finally, our comparative analysis presents that NIOM-DGA significantly outperforms the related approaches proposed in the recent literature. The use of Nature-Inspired Algorithms (NIAs) and feature selection and hyperparameter optimisation has demonstrated significant improvements in DGA detection accuracy and generalisability. This highlights the transformative potential of NIAs in enhancing traditional machine learning algorithms for more robust and effective detection.

Deploying NIOM-DGA in real-world scenarios could face several challenges. The model's reliance on large datasets and feature optimisation may introduce significant computational overhead, limiting scalability in resource-constrained environments. Additionally, as DGA techniques evolve, the model may require frequent updates and retraining to maintain its detection accuracy. Integrating NIOM-DGA into existing systems like SIEM, EDR, XDR, or cloud security platforms could be complex, requiring customisation to align with specific workflows. Real-time domain name analysis may also introduce latency, which could affect immediate threat detection. Furthermore, the model's performance might vary depending on the diversity of data across different organisations, requiring fine-tuning for specific environments.

In contracts, to address these challenges, several solutions can be implemented. To manage computational overhead, optimising the feature selection process and leveraging hardware acceleration, such as GPUs or distributed computing, could enhance scalability and efficiency. To tackle the adaptability to evolving DGAs, the model could incorporate continual learning mechanisms, allowing it to adapt to new DGA patterns through periodic updates and incremental training with fresh data. For integration complexity, designing modular interfaces and offering pre-built integration options with common security platforms would simplify deployment and reduce customisation efforts. To mitigate latency concerns, optimising the model for faster inference times and implementing a hybrid approach that combines pre-processing with real-time analysis could help maintain performance without sacrificing speed. Finally, to address the challenge of data diversity, the model could be trained on a broader range of datasets and offer configuration options to tailor it to specific organisational environments, ensuring better accuracy and adaptability.

Future work in this domain could delve deeper into adversarial attacks and defence strategies. Exploring Generative Adversarial Networks (GANs) to generate adversarial examples could uncover model weaknesses, leading to more robust defence mechanisms. Adversary retraining strategies could be further investigated to enhance model resilience against attacks. In contrast, novel techniques like ensemble learning and model distillation offer additional avenues for improving model security and resilience.

8. Abbreviations

This research uses various abbreviations to describe concepts, techniques, and algorithms relevant to the study's objectives, as shown in Table 14.

Dataset used for this research

The cleaned dataset that was used for features engineering can be downloaded here NIOM-DGA-Research.

Extrahop network dataset

This dataset comprises 16 million Domain Generation Algorithm (DGA) and benign domains. The dataset is accessible through the following link: ExtrahopNetworkDataset (Extrahop Network, 2024).

Table 14
Abbreviations and definitions.

Acronym	Definition	Acronym	Definition
AdaBoost	Adaptive Boosting	ExtraTrees	Extremely Randomised Trees
APC	Asynchronous Procedure Call	FA	Firefly Algorithm
BA	Bat Algorithm	FN	False Negative
BiLSTM	Bidirectional Long Short-Term Memory	FP	False Positive
C&C/C2	Command and Control Server	GANs	Generative Adversarial Networks
CNN	Convolutional Neural Network	GB	Gradient Boosting
CSV	Comma-Separated Values	GWO	Grey Wolf Optimiser
EDR	Endpoint Detection Response	SIEM	Security Information and Event Management
DLL	Dynamic-Link Library	IDS	Intrusion Detection System
DGA	Domain Generation Algorithm	LSTM	Long Short-Term Memory
DNS	Domain Name System	LR	Logistic Regression
DT	Decision Trees	MLP	Multi-Layer Perceptron
RF	Random Forest	ML	Machine Learning
RNG	Random Number Generation	NIA	Nature-Inspired Algorithm
SHAP	SHapley Additive exPlanation	NLP	Natural Language Processing
SLD	Second-Level Domains	RNN	Recurrent Neural Network
TLD	Top-Level Domains	RAT	Remote Access Trojans
TN	True Negative	XAI	eXplainable Artificial Intelligence
TP	True Positive	XGB	eXtreme Gradient Boosting

CRediT authorship contribution statement

Daniel Jeremiah: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Husnain Rafiq:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Conceptualization. **Vinh Thong Ta:** Writing – review & editing, Supervision, Project administration, Methodology. **Muhammad Usman:** Writing – review & editing, Supervision, Project administration, Methodology. **Mohsin Raza:** Writing – review & editing, Supervision, Project administration, Methodology. **Muhammad Awais:** Writing – review & editing, Supervision, Project administration, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Abakumov, A., 2024. DGA: Algorithms for generating domain names and dictionaries of malicious domain names. https://github.com/andrewaeva/DGA. (Accessed 09 May 2025).

Abhiram, P., Anver, S.R., Abdul Rahiman, M., 2023. A deep learning framework for domain generation algorithm based malware detection. https://www.researchsquare.com/article/rs-3154412/v1.

Affinito, A., Zinno, S., Stanco, G., Botta, A., Ventre, G., 2023. The evolution of Mirai botnet scans over a six-year period. J. Inf. Secur. Appl. 79, 103629.

Ahmed, J., Gharakheili, H.H., Russell, C., Sivaraman, V., 2022. Automatic detection of DGA-enabled malware using SDN and traffic behavioral modeling. IEEE Trans. Netw. Sci. Eng. 9, 2922–2939. http://dx.doi.org/10.1109/tnse.2022.3173591.

Anderson, H.S., Woodbridge, J., Filar, B., 2016. DeepDGA: Adversarially-tuned domain generation and detection. In: Proc. 2016 ACM Work. on Artif. Intell. Secur.. http://dx.doi.org/10.1145/2996758.2996767.

Ashley, D., Hindarto, D., 2015. An analysis of gameover zeus network traffic. https://www.giac.org/paper/gcia/8038/analysis-gameover-zeus-network-traffic/114651.

Avertium, 2022. Everything you need to know about bumblebee malware. https://explore.avertium.com/resource/everything-you-need-to-know-about-bumblebee-malware.

Bader, J., 2022. The domain generation algorithm of orchard v3 - a DGA seeded by the bitcoin genesis block. https://bin.re/blog/a-dga-seeded-by-the-bitcoin-genesis-block/. (Accessed 11 May 2025).

Bader, J., 2023. BumbleBee (malware family). https://bin.re/blog/the-dga-of-bumblebee/.

Bader, J., 2024. baderj_domain_generation_algorithms. https://github.com/baderj/domain_generation_algorithms/tree/master.

Basak, J., et al., 2023. On computing the jaro similarity between two strings. In: Lect. Notes Comput. Sci., pp. 31–44. http://dx.doi.org/10.1007/978-981-99-7074-2_3.

Ben, H., 2024. Risks to DNS security and how to uncover DNS based attacks with exabeam. https://community.exabeam.com/s/article/Risks-to-DNS-Security-and-How-to-Uncover-DNS-Based-Attacks-with-Exabeam.

Brandstaetter, S., 2024. Protecting against cyber threats: the use of domain generation algorithm (DGA) by threat actors. https://osintph.medium.com/protecting-against-cyber-threats-the-role-of-domain-generation-algorithm-dga-80c3ec3cda9f. (Accessed 25 April 2024).

Camacho Villalón, C.L., Stützle, T., Dorigo, M., 2020. Grey wolf, firefly and bat algorithms: Three widespread algorithms that do not contain any novelty. Springer Nat. Switz. AG 2020 121–133. http://dx.doi.org/10.1007/978-3-030-60376-2_10.

Chagneau, A., Massaoudi, Y., Derbali, I., Yahiaoui, L., 2024. Quantum algorithm for bioinformatics to compute the similarity between proteins. arXiv preprint arXiv:2402.09927.

Chen, J.-L., Qiu, J.-F., Chen, Y.-H., 2024. A hybrid DGA DefenseNet for detecting DGA domain names based on FastText and deep learning techniques. Comput. Secur. 150, 104232. http://dx.doi.org/10.1016/j.cose.2024.104232.

Corley, I., Lwowski, J., Hoffman, J., 2020. DomainGAN: Generating adversarial examples to attack domain generation algorithm classifiers. https://arxiv.org/abs/1911.06285.

Daji, Suqitian, 2022. DGA family orchard continues to change; the new version uses Bitcoin transaction information to generate DGA domain names. 360 Netlab Blog - Network Security Research Lab at 360 [online]. (Accessed 6 September 2024).

DGArchive, 2025. DGArchive - Fraunhofer FKIE: DGArchive a database of domain names generated by domain generation algorithms (DGAs), most often found in malware. https://dgarchive.caad.fkie.fraunhofer.de/. (Accessed 09 May 2025).

Ding, L., Du, P., Hou, H., Zhang, J., Jin, D., Ding, S., 2023. Botnet DGA domain name classification using transformer network with hybrid embedding. Big Data Res. 33, 100395. http://dx.doi.org/10.1016/j.bdr.2023.100395.

El-Ghamry, A., Gaber, T., Mohammed, K.K., Hassanien, A.E., 2023. Optimized and efficient image-based IoT malware detection method. Electronics 12 (3), 708. http://dx.doi.org/10.3390/electronics12030708.

Elizarov, D., Katkov, A., 2023. Practical malware analysis. Her. Dagestan State Tech. Univ. Tech. Sci. 50 (3), 66–71.

Extrahop Network, 2024. ExtraHop/DGA-Detection-Training-Dataset. https://github.com/ExtraHop/DGA-Detection-Training-Dataset.

Ford, N., 2024. Global data breaches and cyber attacks in 2024. https://www.itgovernance.co.uk/blog/global-data-breaches-and-cyber-attacks-in-2024. (Accessed 01 May 2024).

Givre, C., DGA Dataset, https://www.kaggle.com/datasets/gtkcyber/dga-dataset.

Gogoi, B., Ahmed, T., 2023. DGA domain detection using pretrained character based transformer models. In: 2023 IEEE Guwahati Subsection Conference. GCON, pp. 01–06. http://dx.doi.org/10.1109/GCON58516.2023.10183602.

Gregório, R., 2023. DGA_Legit_ASCII: DGA Netlab Majestic Legit. https://www.kaggle.com/datasets/rafaelgregrio/dga-legit-ascii. (Accessed 09 May 2025).

Griffiths, C., 2023. The latest ransomware statistics (updated january 2023) | AAG IT support. https://aag-it.com/the-latest-ransomware-statistics/.

Harpo, 2023. Harpomaxx: DGA-detection dataset. https://huggingface.co/datasets/harpomaxx/dga-detection. (Accessed 09 May 2025).

Hasan, M.R., Ferdous, J., 2024. Dominance of AI and machine learning techniques in hybrid movie recommendation system applying text-to-number conversion and cosine similarity approaches. J. Comput. Sci. Technol. Stud. 6 (1), 94–102.

Hassaoui, M., Hanini, M., El Kafhali, S., 2024. Unsupervised clustering for a comparative methodology of machine learning models to detect domain-generated algorithms based on an alphanumeric features analysis. J. Netw. Syst. Manage. 32 (1), http://dx.doi.org/10.1007/s10922-023-09793-6.

Highnam, K., Puzio, D., Luo, S., Jennings, N.R., 2021. Real-time detection of dictionary DGA network traffic using deep learning. SN Comput. Sci. 2, http://dx.doi.org/10.1007/s42979-021-00507-w.

Hoang, X.D., Vu, X.H., 2021. An improved model for detecting DGA botnets using random forest algorithm. Inf. Secur. J.: A Glob. Perspect. 1–10. http://dx.doi.org/10.1080/19393555.2021.1934198.

Hsu, C.-M., Yang, C.-C., Cheng, H.-H., Setiasabda, P.E., Leu, J.-S., 2021. Enhancing file entropy analysis to improve machine learning detection rate of ransomware. IEEE Access 9, 138345–138351. http://dx.doi.org/10.1109/ACCESS.2021.3114148.

Hu, X., Chen, H., Li, M., Cheng, G., Li, R., Wu, H., Yuan, Y., 2023. ReplaceDGA: BiLSTM based adversarial DGA with high anti-detection ability. IEEE Trans. Inf. Forensics Secur..

Hu, X., Li, M., Cheng, G., Li, R., Wu, H., Gong, J., 2022. Towards accurate DGA detection based on siamese network with insufficient training samples. In: ICC 2022 - IEEE Int. Conf. on Commun. http://dx.doi.org/10.1109/icc45855.2022.9838409.

Hu, B., Wang, H., Wu, K., Li, Z., 2024. Automatic assessing system of operating process based on Needleman-Wunsch algorithm. Int. Core J. Eng. 10 (3), 205–212.

Javaheri, D., Gorgin, S., Lee, J.-A., Masdari, M., 2023. Fuzzy logic-based ddos attacks and network traffic anomaly detection methods: Classification, overview, and future perspectives. Inform. Sci. 626, 315–338.

Javed, M.A., Rashid, I., Rashdi, A., 2023. DGA malware deep learning detection and its optimization with novel activation function. J. Comput. Biomed. Inform. 4, 285–297.

Kaspersky, 2025. Kaspersky state of ransomware report–2025: Global and regional insights for international anti-ransomware day. https://www.kaspersky.com/about/press-releases/kaspersky-state-of-ransomware-report-2025-global-and-regional-insights-for-international-anti-ransomware-day. (Accessed 09 May 2025).

Katz, O., 2021. Digging deeper – an in-depth analysis of a fast flux network. https://www.akamai.com/blog/security/digging-deeper-an-in-depth-analysis-of-a-fast-flux-network.

Kemmerling, T., 2023. Dataset for detecting domains generated by algorithm | ExtraHop. https://www.extrahop.com/blog/dataset-for-detecting-domains-generated-by-algorithm-extrahop. (Accessed 01 May 2024).

Kirişci, M., 2023. New cosine similarity and distance measures for fermatean fuzzy sets and TOPSIS approach. Knowl. Inf. Syst. 65 (2), 855–868.

Kostopoulos, N., Kalogeras, D., Pantazatos, D., Grammatikou, M., Maglaris, V., 2023. SHAP interpretations of tree and neural network DNS classifiers for analyzing DGA family characteristics. IEEE Access 11, 61144–61160. http://dx.doi.org/10.1109/access.2023.3286313.

Liang, J., Chen, S., Wei, Z., Zhao, S., Zhao, W., 2022. HAGDetector: Heterogeneous DGA domain name detection model. Comput. Secur. 120, 102803. http://dx.doi.org/10.1016/j.cose.2022.10280.

Likić, V., 2007. The Needleman-Wunsch algorithm for sequence alignment. 7th Melb. Bioinform. Course 1/46.

Mehri, B., Goussard, Y., Trépanier, M., 2023. Lookup Table String Similarity Algorithm. Bureau de Montreal, Université de Montreal.

Mehta, G., Jain, S., Das, P., Tripathi, V., 2023. Machine learning approach for malware detection and classification using bio inspired algorithms. In: 2023 International Conference on Evolutionary Algorithms and Soft Computing Techniques. EASCT, pp. 1–6. http://dx.doi.org/10.1109/EASCT59475.2023.10392768.

Mestour, Z., 2021. Alexa 1millioon and domain generation algorithm. https://www.kaggle.com/datasets/slashtea/domain-generation-algorithm.

Moşolea, V., Oprişa, C., 2023. Detecting domain generation algorithms in malware traffic using constrained resources. http://dx.doi.org/10.1109/iccp60212.2023.10398684, IEEE.

N., M., D., R., S., M., Sharma, V., 2022. Performance analysis of DGA-driven botnets using artificial neural networks. In: 2022 10th Int. Conf. on Reliab. Infocom Technol. Optim. (Trends Futur. Dir.. ICRITO, http://dx.doi.org/10.1109/icrito56286.2022.9965044.

Nowroozi, E., Mekdad, Y., Berenjestanaki, M.H., Conti, M., EL Fergougui, A., 2022. Demystifying the transferability of adversarial attacks in computer networks. IEEE Trans. Netw. Serv. Manag. http://dx.doi.org/10.1109/tnsm.2022.3164354, 1–1.

Pan, R., Chen, J., Ma, H., Bai, X., 2022. Using extended character feature in Bi-LSTM for DGA domain name detection. IEEE Access http://dx.doi.org/10.1109/icis54925.2022.9882343.

Papadogiannaki, E., Ioannidis, S., 2023. Pump up the JARM: Studying the evolution of botnets using active TLS fingerprinting. In: 2023 IEEE Symposium on Computers and Communications. ISCC, IEEE, pp. 764–770.

Patil, S.D., Dharme, M., Patil, A.J., Gautam, Jarial, R.K., Singh, A., 2022. DGA based ensemble learning and random forest models for condition assessment of transformers. IEEE Access http://dx.doi.org/10.1109/smartgencon56628.2022.10083672.

Patsakis, C., Casino, F., 2021. Exploiting statistical and structural features for the detection of domain generation algorithms. J. Inf. Secur. Appl. 58, 102725. http://dx.doi.org/10.1016/j.jisa.2020.102725.

Prasetya, A.Y., Aini, K.I., Lim, C., 2023. Comparative analysis of attack behavior patterns in petya, cryptinfinite, and locky ransomware using hybrid analysis. In: 2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity. ICoCICs, IEEE, pp. 29–34.

Pye, J., Issac, B., Aslam, N., Rafiq, H., 2020. Android malware classification using machine learning and bio-inspired optimisation algorithms. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 1777–1782. http://dx.doi.org/10.1109/TrustCom50675.2020.00244.

Quezada, V., Astudillo-Salinas, F., Tello-Oquendo, L., Bernal, P., 2022. Real-time bot infection detection system using dns fingerprinting and machine-learning. SSRN Electron. J. http://dx.doi.org/10.2139/ssrn.4292617.

Rafiq, H., Aslam, N., Aleem, M., Issac, B., Randhawa, R.H., 2022. AndroMalPack: Enhancing the ML-based malware classification by detection and removal of repacked apps for android systems. Sci. Rep. 12 (1), http://dx.doi.org/10.1038/s41598-022-23766-w.

Rana, S., Aksoy, A., 2021. Automated fast-flux detection using machine learning and genetic algorithms. In: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops. INFOCOM WKSHPS, pp. 1–6. http://dx.doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484614.

Randhawa, R.H., Aslam, N., Alauthman, M., Rafiq, H., 2023. Evasion generative adversarial network for low data regimes. IEEE Trans. Artif. Intell. 4 (5), 1076–1088. http://dx.doi.org/10.1109/tai.2022.3196283.

Randhawa, R.H., Aslam, N., Alauthman, M., Rafiq, H., Comeau, F., 2021. Security hardening of botnet detectors using generative adversarial networks. IEEE Access 9, 78276–78292. http://dx.doi.org/10.1109/access.2021.3083421.

Rao, M.V., Midhunchakkaravarthy, D., Dandu, S., 2023. Propagation of computer worms—A study. In: International Conference on Soft Computing and Signal Processing. Springer, pp. 629–639.

Ren, Y., Li, H., Liu, P., Liu, J., Zhu, H., Sun, L., 2023. CL-GAN: a GAN-based continual learning model for generating and detecting AGDs. Comput. Secur. 131, 103317. http://dx.doi.org/10.1016/j.cose.2023.103317.

Sahni, S., Rajasekaran, S., 2023. On computing the jaro similarity between two strings. In: Bioinformatics Research and Applications: 19th International Symposium, ISBRA 2023, WrocłAw, Poland, October 9–12, 2023, Proceedings, vol. 14248, Springer Nature, p. 31.

Sea, Law, N.F., 2023. Use of subword tokenization for domain generation algorithm classification. Cybersecurity 6, http://dx.doi.org/10.1186/s42400-023-00183-8.

Shah, A.V., 2025. 15 million domain names dataset: DGA and benign domains. https://www.kaggle.com/datasets/aayushah19/dga-or-benign-domain-names. (Accessed 09 May 2025).

Shahzad, H., Sattar, A.R., Skandaraniyam, J., 2021. DGA domain detection using deep learning. In: 2021 IEEE 5th Int. Conf. on Cryptogr. Secur. Priv.. CSP, http://dx.doi.org/10.1109/csp51677.2021.9357591.

Sharma, G., 2023. ExtraHop open sources its machine learning dataset. https://itbrief.co.uk/story/extrahop-open-sources-its-machine-learning-dataset. (Accessed 01 May 2024).

Sharma Paudel, S., 1.5 Million (Umbrella, DGA) Domains with Classes, https://www.kaggle.com/datasets/sabindcoster/15-million-umbrelladga-domains-with-classes.

Sidi, L., Nadler, A., Shabtai, A., 2019. MaskDGA: a black-box evasion technique against DGA classifiers and adversarial defenses. http://dx.doi.org/10.48550/arXiv.1902.08909.

Sivaguru, R., Peck, J., Olumofin, F., Nascimento, A., De Cock, M., 2020. Inline detection of DGA domains using side information. IEEE Access 8, 141910–141922. http://dx.doi.org/10.1109/access.2020.3013494.

Sreekanta, N., 2022. Machine learning in security: Deep learning based DGA detection with a pre-trained model. https://www.splunk.com/en_us/blog/security/machine-learning-in-security-deep-learning-based-dga-detection-with-a-pre-trained-model.html.

Sun, X., Liu, Z., 2023. Domain generation algorithms detection with feature extraction and domain center construction. PLoS One 18, e0279866. http://dx.doi.org/10.1371/journal.pone.0279866.

Suryotrisongko, H., Musashi, Y., 2022. Evaluating hybrid quantum-classical deep learning for cybersecurity botnet DGA detection. Procedia Comput. Sci. 197, 223–229. http://dx.doi.org/10.1016/j.procs.2021.12.135.

Sutheekshan, B., Basheer, S., Thangavel, G., Sharma, O.P., 2024. Evolution of malware targeting IoT devices and botnet formation. IC2PCT, In: 2024 IEEE International Conference on Computing, Power and Communication Technologies, vol. 5, IEEE, pp. 1415–1422.

Tatar, O., 2024. DGA data sets. https://www.kaggle.com/datasets/omurcantatar/dga-data-sets. (Accessed 09 May 2025).

Tuan, T.A., Long, H.V., Taniar, D., 2022. On detecting and classifying DGA botnets and their families. Comput. Secur. 113, 102549. http://dx.doi.org/10.1016/j.cose.2021.102549.

Tyagi, A., 2023. International journal of creative research thoughts (IJCRT). Int. J. Creative Res. Thoughts (IJCRT): Int. Open Access Peer- Rev. Ref. J. 11 (9), 2320–2882.

Velasco-Mata, J., González-Castro, V., Fidalgo, E., Alegre, E., 2023. Real-time botnet detection on large network bandwidths using machine learning. Sci. Rep. 13, http://dx.doi.org/10.1038/s41598-023-31260-0.

Wang, Z., Guo, Y., 2021. Neural networks based domain name generation. J. Inf. Secur. Appl. (ISSN: 2214-2126) 61, 102948. http://dx.doi.org/10.1016/j.jisa.2021.102948.

Wang, H., Tang, Z., Li, H., Zhang, J., Li, S., Wang, J., 2023. CI_GRU: An efficient DGA botnet classification model based on an attention recurrence plot. Comput. Netw. 235, 109992.

Wang, S., Zhou, Z., Li, B., Li, Z., Kan, Z., 2024. Multi-modal interaction with transformers: bridging robots and human with natural language. Robotica 42 (2), 415–434.

Weissgerber, N., Jenke, T., Padilla, E., Bruckschen, L., 2023. Open SESAME: Fighting botnets with seed reconstructions of domain generation algorithms. http://dx.doi.org/10.48550/arXiv.2301.05048.

Yan, L., Yang, Y., Li, Y., Wang, W., Yu, Z.Y., 2023. Classification of malicious DGA domain name families based on BiGRU and attention mechanisms. In: IEEE ITAIC. http://dx.doi.org/10.1109/itaic58329.2023.10408904, ISSN: 2693-2865.

Zhao, D., Li, H., Sun, X., Tang, Y., 2023a. Detecting DGA-based botnets through effective phonics-based features. Future Gener. Comput. Syst. 143, 105–117. http://dx.doi.org/10.1016/j.future.2023.01.027.

Zhao, D., Li, H., Sun, X., Tang, Y., 2023b. Detecting DGA-based botnets through effective phonics-based features. Future Gener. Comput. Syst. 143, 105–117.