future internet

Article

Distributed Denial of Service Classification for Software-Defined Networking Using Grammatical Evolution

Evangelos D. Spyrou [1], Ioannis Tsoulos [1,*] and Chrysostomos Stylios [1,2]

[1] Department of Informatics and Telecommunications, University of Ioannina, 47150 Arta, Greece; e.spyrou@uoi.gr (E.D.S.); stylios@uoi.gr (C.S.)
[2] Industrial Systems Institute, Athena Research Center, 26504 Patras, Greece
* Correspondence: itsoulos@uoi.gr

Abstract: Software-Defined Networking (SDN) stands as a pivotal paradigm in network implementation, exerting a profound influence on the trajectory of technological advancement. The critical role of security within SDN cannot be overstated, with distributed denial of service (DDoS) emerging as a particularly disruptive threat, capable of causing large-scale disruptions. DDoS operates by generating malicious traffic that mimics normal network activity, leading to service disruptions. It becomes imperative to deploy mechanisms capable of distinguishing between benign and malicious traffic, serving as the initial line of defense against DDoS challenges. In addressing this concern, we propose the utilization of traffic classification as a foundational strategy for combatting DDoS. By categorizing traffic into malicious and normal streams, we establish a crucial first step in the development of effective DDoS mitigation strategies. The deleterious effects of DDoS extend to the point of potentially overwhelming networked servers, resulting in service failures and SDN server downtimes. To investigate and address this issue, our research employs a dataset encompassing both benign and malicious traffic within the SDN environment. A set of 23 features is harnessed for classification purposes, forming the basis for a comprehensive analysis and the development of robust defense mechanisms against DDoS in SDN. Initially, we compare GenClass with three common classification methods, namely the Bayes, K-Nearest Neighbours (KNN), and Random Forest methods. The proposed solution improves the average class error, demonstrating 6.58% error as opposed to the Bayes method error of 32.59%, KNN error of 18.45%, and Random Forest error of 30.70%. Moreover, we utilize classification procedures based on three methods based on grammatical evolution, which are applied to the aforementioned data. In particular, in terms of average class error, GenClass exhibits 6.58%, while NNC and FC2GEN exhibit average class errors of 12.51% and 15.86%, respectively.

Keywords: SDN; DDoS; genetic algorithm; grammatical evolution; packet classification

check for updates

Citation: Spyrou, E.D.; Tsoulos, I.; Stylios, C. Distributed Denial of Service Classification for Software-Defined Networking Using Grammatical Evolution. *Future Internet* **2023**, *15*, 401. https://doi.org/10.3390/fi15120401

Academic Editor: Franco Davoli

Received: 14 November 2023
Revised: 1 December 2023
Accepted: 5 December 2023
Published: 13 December 2023

1. Introduction

Recent advancements in Information and Communication Technology (ICT), encompassing big data, cloud computing, mobile technologies, and multimedia, have prompted a growing need for enhanced service management, an increased user bandwidth, and improved Internet accessibility. To address these evolving demands, Software-Defined Networking (SDN) has emerged as a promising solution. A comprehensive overview of SDN, including its essential features, is provided by [1,2]. This summary underscores two key characteristics of SDN: the programmability of the control plane and the separation of control and data planes. However, it is emphasized that these aspects, as elaborated in the subsequent discussion, are not entirely novel in the realm of network architecture.

In the domain of SDN, its distinctive feature lies in providing programmability through the clear separation of the control and data planes. This approach fundamentally transforms the way network devices are programmed, offering a user-friendly alternative to the intricacies of active networking. SDN advocates for the separation of control and data planes within the network's architectural framework, enabling the autonomous management of network control on the control plane without disrupting the data flow. This configuration facilitates the extraction of network intelligence from switching devices, relocating it to controllers. Simultaneously, it empowers external software to manage switching devices without necessitating embedded intelligence.

The decoupling of the control plane from the data plane not only establishes a simplified, programmable environment but also provides increased flexibility for external software to shape the network's behavior. This transformation signifies a pivotal shift in network management and control, streamlining the process and creating opportunities for a more adaptable and responsive network infrastructure in the context of SDN.

1.1. Motivation

In recent years, there has been a growing interest within the scientific community to explore the realm of SDN security, aiming to overcome barriers hindering widespread SDN adoption. This research domain has a dual focus: one facet aims to harness SDN features to enhance security, while the other endeavors to establish a secure architecture for SDN systems. As outlined in [3], the paper offers an overview of security threats that pose risks to SDN and enumerates various attacks exploiting vulnerabilities and misconfigurations in the constituent elements of SDN. Additionally, the paper delves into a discussion highlighting the duality between utilizing SDN for security improvement and securing SDN itself. SDN security in IoT networks has emerged as well and it is presented in [4].

A distributed denial of service (DDoS) attack constitutes a malicious attempt to disrupt the regular flow of data to a specific server, service, or network by inundating the target or its surrounding infrastructure with an overwhelming surge of Internet traffic [5]. These attacks achieve their objectives by harnessing multiple compromised computer systems, encompassing both traditional computers and networked resources, such as Internet of Things (IoT) devices or SDN devices [6,7], to generate attack traffic. In a broader context, visualizing a DDoS attack is akin to an unexpected traffic bottleneck on a network, obstructing the regular flow of traffic to its intended destination. The impact is analogous to an overwhelming surge of vehicles congesting a road, preventing smooth passage to the intended locations.

DDoS attacks present a significant and escalating threat to the Internet [8]. Attackers continually adapt their tactics to evade security systems, prompting researchers to consistently refine their approaches to counter new attack strategies. As a result, the DDoS landscape has become increasingly advanced, making it challenging to attain a comprehensive understanding of the situation. On one hand, this complexity impedes a clear comprehension of the DDoS phenomenon, given the multitude of known attack types that create the impression of a vast and challenging problem. On the other hand, the diverse strategies employed by existing defense systems further complicate matters, making it challenging to identify their commonalities and differences and assess their effectiveness [5]. Previous works in the literature have explored the intersection of DDoS and SDN, as evident in [9–12].

There exist very interesting pieces of research that highlight workflow scheduling in SDN-based IoT and Fog networks [13,14] and mitigation strategies in IoT systems [15]. Leveraging SDN enables the routing of diverse network traffic through a unified physical network infrastructure while maintaining the desired level of isolation. To counter undesirable malicious traffic, an SDN-based IoT access control application is deployed on the SDN controller. This application meticulously examines each incoming/outgoing connection to IoT domains in accordance with predefined security policies. Approved connections trigger the SDN controller to issue relevant forwarding rules in the physical/virtual SDN switches along the specified networking path. Conversely, malicious traffic from cybercriminals is thwarted by implementing specific network access lists. Moreover, it has the capability to dynamically segregate malicious or suspicious network flows. SDN-based separation solutions, in this context, can provide varying levels of network abstractions. This allows for the effective separation of network traffic and the presentation of network views aligned with the desired security properties [16].

Additionally, the SDN controller possesses extensive visibility into the supervised data planes and, via the control plane, gathers network status information by dispatching statistics query messages to the switches. This enables the SDN controller to provide real-time updates on the underlying infrastructure and relay flow request messages to network applications operating on the control plane. Such an approach significantly streamlines the development of strategies for implementing anomaly network analysis and the detection of network-wide attacks. A critical facet of SDN lies in the dynamic installation and updating of forwarding rules by the SDN controller in network elements, facilitating efficient traffic flow management. This increased manageability markedly amplifies the potential for network applications to implement tailored and effective security mechanisms [16]. As such, we require a mechanism that classifies network traffic to identify the anomalies produced by malicious users or bots.

1.2. Contributions

In our research, we delve into the intricate domain of Software-Defined Networking (SDN) by analyzing a diverse dataset that includes various network traffic patterns, ranging from benign to malicious. This dataset consists of 23 distinct features, forming the foundation for our classification efforts. To tackle the classification task within the SDN dataset, we adopt grammatical evolution-based methodologies [17,18]. This innovative approach leverages evolutionary algorithms to discern complex patterns and interactions inherent in the data, offering a unique perspective on the challenge of classifying network traffic in the context of Software-Defined Networking.

This research concludes with the presentation of outcomes that provide valuable insights into the effectiveness of the chosen approach, centered around grammatical evolution-based classification methods. The results not only illuminate the performance of these methods but also contribute significantly to the broader understanding of SDN security and the classification of network traffic in this dynamic field. The motivation behind this work stems from the increasing frequency of attacks in networked systems, particularly distributed denial of service (DDoS) attacks, which pose a severe threat by potentially causing system failures rather than just faults.

More specifically, the contributions of this paper are the following:

- Initially, we compare GenClass with three classification methods, namely the Bayes, K-Nearest Neighbors (KNN), and Random Forest methods, and demonstrate that our solution improves the average class error as opposed to the competitors.
- We encapsulate three methods based on grammatical evolution and show that GenClass exhibits a lower average class error than NNC and FC2GEN.
- We demonstrate that the three grammatical evolution methods exhibit low average class error values.

2. Related Work

There is a plethora of works on defense of DDoS attacks in SDN using machine learning mechanisms [19–22]. A more detailed literature review follows.

An important survey on mitigation techniques regarding DDoS attacks in SDN is given in [23]. The authors systematically categorized their research into two key domains: one focusing on strategies to counter denial of service (DoS) attacks within Software-Defined Networking (SDN) and the other exploring SDN-centric methods to counteract DoS attacks across diverse networks. In the first category of solutions, they identified six distinct classifications: table-entry, scheduling, architectural, flow statistics, machine learning, and hybrid solutions. Furthermore, the authors conducted a thorough examination of the tools and datasets featured in the reviewed contributions. It delivers a detailed comparative analysis of these approaches, taking into account factors such as the engagement of network devices, network layers, types of DoS attacks, and the specific targets subjected to these attacks.

In the paper referenced in [24], a comprehensive analysis of approximately 70 established techniques designed for detecting and mitigating distributed denial of service (DDoS) attacks within Software-Defined Networking (SDN) environments is conducted. These techniques are systematically categorized into four primary groups, incorporating methods based on information theory, machine learning, Artificial Neural Networks (ANNs), and various miscellaneous approaches. Additionally, the paper extensively explores and addresses persistent research challenges, gaps, and issues associated with establishing a secure DDoS defense solution in the realm of SDN. This detailed review is poised to serve as a valuable resource for the research community, aiding the development of more robust and reliable DDoS mitigation solutions tailored for SDN networks.

In [25], the paper proposes leveraging the central control features of SDN for attack detection, introducing an efficient and resource-aware solution. Specifically, the paper delves into how DDoS attacks can strain controller resources and presents a method for identifying these attacks by analyzing the entropy variation in the destination IP address. Notably, this approach demonstrates the ability to detect DDoS attacks within the first five hundred packets of the attack traffic. This early detection capability is a significant advancement in proactively identifying and mitigating DDoS threats, enhancing the overall security posture of SDN environments.

In [26], the authors highlight the limitations of traditional methods reliant on fixed thresholds and historical data, inhibiting their adaptability to new and evolving DDoS attack scenarios. They propose an innovative approach for detecting DDoS attacks within Software-Defined Networking (SDN) environments. This novel method incorporates three vital components: a collector, an entropy-based module, and a classification stage. Extensive experiments utilizing UNB-ISCX, CTU-13, and ISOT datasets demonstrate that this approach surpasses existing methods in terms of accuracy for DDoS attack detection in SDN environments.

Moving to [27], the authors conducted a comprehensive evaluation of the latest advancements in machine learning (ML) and deep learning (DL) methodologies for detecting distributed denial of service (DDoS) attacks within SDN contexts. Their work involved an extensive systematic review focusing on publications utilizing ML/DL techniques to uncover DDoS attacks in SDN networks spanning from 2018 through to early November 2022. This evaluation provides a valuable and updated insight into the evolution and effectiveness of ML and DL techniques in combating DDoS threats within SDN environments.

In [28], the authors introduce a DDoS attack detection and defense system that leverages cognitive-inspired computing along with dual address entropy. This system involves extracting attributes from the switch's flow table, creating a DDoS attack model using the support vector machine classification algorithm, and enabling real-time detection and defense in the initial stages of a DDoS attack, ensuring the swift restoration of regular communication. Their findings emphasize the system's rapid attack detection, high accuracy in detection, and a low rate of false positives. Moreover, it is capable of initiating appropriate defense and recovery measures upon identifying an attack.

In another study, the authors [29] utilize the Neighborhood Component Analysis (NCA) algorithm for feature selection and an effective classification phase. After preprocessing and feature selection, they employ the K-Nearest Neighbor (KNN), Decision Tree (DT), Artificial Neural Network (ANN), and Support Vector Machine (SVM) algorithms on a similar dataset. Their experimental results reveal the DT algorithm's superior performance, achieving an impressive 100% classification accuracy rate when compared to other algorithms.

Brief Comparison

The aforementioned research works present some interesting methods for the detection and mitigation of DDoS traffic in SDN networks. The approach that is being proposed in this paper aims to show the efficiency of simple genetic algorithms towards the detection and classification of malicious traffic. Some of the suggested related works may increase the complexity of the detection and not provide a simple technique. As will be evident in the results part of this paper, simple algorithms will exhibit a worse average class error value than the proposed algorithm.

3. Proposed Approach

Here, we provide the proposed work with the reference architecture to the reader.

3.1. Reference Architecture and Problem Statement

Distributed denial of service (DDoS) attacks involve orchestrating Internet-enabled devices, including traditional computers and SDN devices, into a network called a "botnet". Compromised devices within the botnet, referred to as "bots" or "zombies", are controlled remotely by an attacker. The attacker coordinates the assault by instructing each bot to send numerous requests to overwhelm a target server or network, causing a denial of service.

Mitigating DDoS attacks is challenging as bots mimic legitimate devices, making it difficult to distinguish attack traffic from regular data. Indications of a DDoS attack include a sudden decrease in website or service speed, or even complete unavailability. Traffic analysis tools are essential for discerning specific indicators.

Specialized signs of a DDoS attack vary based on the attack type, encompassing application-layer, protocol-based, and volumetric assaults. Recognizing these signs is crucial for implementing effective defense strategies against diverse modes of DDoS attacks. A visual representation illustrating the nature of a DDoS attack is depicted in Figure 1.

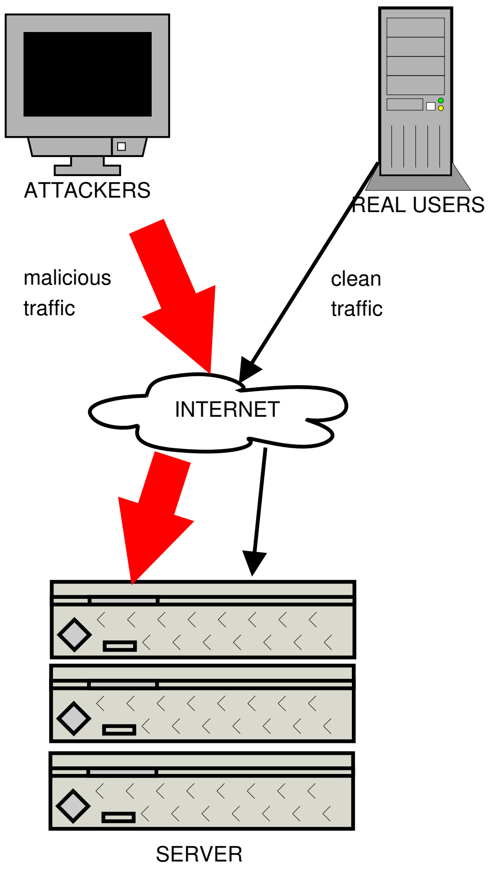

Figure 1. DDoS scheme.

In the figure, we can see that the server is overloaded with malicious data, not allowing the genuine data to reach the location. The early detection of malicious traffic is of primary importance to SDN. In this way, the benign traffic will be able to reach the server without problems. The mitigation strategies include dropping the malicious traffic before it is handled by the SDN device. As such, the necessity of a classifier that performs this operation is mandatory.

3.2. Proposed Methodologies

Here, we provide the reader with three proposed methods using grammatical evolution (GE) to perform the necessary tasks that are required. Before that, we introduce the reader to the nomenclature of the parameters used.

3.2.1. GenClass

The GenClass method [30] has demonstrated a remarkable performance in tackling classification problems. Furthermore, the method's source code is openly available for use in any classification task. Detailed information about the associated software can be found in Anastasopoulos et al.'s pertinent publication [31]. Grammatical evolution in this context relies on the following key components:

The first component is the grammar of the target language, which is expressed in the Backus–Naur form (BNF) format. This grammar is defined as a context-free grammar (CFG), represented as $G = (N, T, S, P)$, where N signifies the set of nonterminal symbols, T signifies the set of terminal symbols, S represents the starting symbol of the grammar, and P is a set that contains production rules. Each production rule takes the form $A \rightarrow a$ or $A \rightarrow aB$, where A and B belong to N and a belongs to T. The second component is the corresponding fitness function.

In grammatical evolution, chromosomes are represented as vectors of integers, where each element within the chromosome corresponds to a production rule from the provided BNF grammar. Each production rule is assigned a unique serial number. The algorithm initiates by commencing with the starting symbol of the grammar and progressively generates program strings by substituting non-terminal symbols with the right-hand side of the selected production rule. The rule selection process involves two steps:

Take the next element from the chromosome and label it as V. Choose the next production rule based on the scheme $Rule = V$ mod R, where R represents the number of production rules applicable to the current non-terminal symbol. The proposed method is given in the Algorithm 1.

3.2.2. Neural Network Construction (NNC)

Tsoulos et al. [32] proposed a method that employs grammatical evolution (GE) for both structuring the network's topology and refining its weights. Their approach involves encoding the network's architecture and weights using a context-free grammar (CFG) in Backus–Naur form (BNF). The paper highlights that using GE for evolving Artificial Neural Networks (ANNs) offers the advantage of easily shaping the search outcomes and results in a concise encoding. However, while GE facilitates the effective shaping of the search process, it appears less suitable for actual vector optimization, specifically in optimizing connection weights. This limitation may lead to issues such as highly destructive variation operators, potentially erasing information acquired during the evolutionary search. The algorithm is given in Algorithm 2.

3.2.3. FC2GEN

This section summarizes the work in [33], which constitutes the third method that we utilized with the dataset using GE. The FSC method, rooted in grammatical evolution, aims to enhance the classification accuracy of a given classifier by generating new features from existing ones. The process involves several key steps:

Data preparation: The dataset is divided into independent train and test sets. The train set is utilized for constructing features, while the test set is used to evaluate these features in the chosen classification method.

Algorithm 1 GenClass algorithm

```
Require: Read Train Data $(x_i, t_i)$
Ensure: $N_G$
Ensure: $N_C$
Ensure: $P_S$
Ensure: $P_M$ Initialize the chromosomes of the population. Every element of each chromo-
  some is initialised randomly in $r_m$.
Ensure: $iter = 1$
  while $i \leq N_g$ do
    (1)
    Create $C_i$ for $g_i$
    Calculate $f_i = \sum_{i=1}^{M} (C(x_i) - t_i)^2$
    In the genetic algorithm's selection process, chromosomes are classified according to
  their fitness. The top-performing $(1 - P_S) \times N_C$ chromosomes, where $N_C$ represents the
  total number of chromosomes, remain unaltered and are preserved for the subsequent
  generation in the population. The remaining chromosomes are then replaced by new
  ones generated during the crossover phase.
    In the execution of the crossover method, $P_S \times N_C$ chromosomes are generated.
  Initially, for every pair of newly produced offspring, two distinct chromosomes (parents)
  are chosen from the existing population using tournament selection. This entails forming
  a subset of more than one $(K > 1)$ randomly selected chromosome, and the chromosome
  with the highest fitness value is selected as the parent. For each parent pair $(z, w)$, two
  new offspring, $\tilde{z}$ and $\tilde{w}$, are generated using the one-point crossover method.
    Conduct the mutation process by iterating through every component of each chromo-
  some. For each component, select a random number, $r$, from the range $[0, 1]$, and modify
  the corresponding chromosome if $r \leq P_M$.
  end while
  $iter = iter + 1$
  if $iter \leq N_G$ then
    goto (1)
  end if
  Obtain $g^*$ and create $C^*$
  Apply $C^*$ to test set
```

Genetic algorithm parameter definition: Parameters such as N_f (determining the number of constructed or selected features from the original set), N_g (total number of chromosomes in the genetic population), L_g (chromosome size), R_s (fraction of unchanged chromosomes in the next generation), and R_m (mutation rate) are defined. The algorithm uses fixed-length chromosomes to restrict the creation of excessively large expressions and decrease the search space.

Grammar definition: A context-free grammar, outlining the possible algebraic expressions of the original feature set, is created. This grammar includes valid arithmetic operations using the original features, limiting the total number of original features (denoted as N) and defining the start symbol (denoted as S).

Chromosome initialization: Each part of every chromosome in the genetic pool is randomly initialized within the range [0, 255].

Fitness evaluation: Evaluation of each chromosome involves assessing its performance based on some fitness criteria.

Feature construction from chromosome parts: The chromosome is divided into N_f equal parts, and each part $(g_i, i = 1, \ldots, N_f)$ is used to construct a feature. Features

$(f_i, i = 1, \ldots, N_f)$ are created from each part g_i through a mapping process. This construction process can be executed in parallel.

Algorithm 2 NNC algorithm

```
Ensure: $N_G$
Ensure: $N_C$
Ensure: $P_S$
Ensure: $P_M$
Ensure: $L_I$
Ensure: $L_c$
Ensure: Initialize the chromosomes as random vectors of integers.
Ensure: $iter = 0$
  (1)
  while $i \leq N_g$ do
      Create $C_i$ with GE
      Calculate $f_i$
      Apply the genetic operations of crossover and mutation.
  end while
  if $iter \% L_I = 0$ then
      Create random $L_C$
      create $L_S$
      while $X_i \in L_S$ do
          select randomly $Y$ from population
          Create an offspring $Z$ of $X_i$ and $Y$ using one point crossover.
          if $f(z) < f_i$ then
              $Xi = Z,$
              $f_i = f(Z)$
              $iter = iter + 1.$
              if $iter > iter_{max}$ then
                  terminate
              else
                  goto (1)
              end if
          end if
      end while
  end if
  Create a neural network for the best chromosome
  Evaluate the neural network.
```

Data transformation based on constructed features: The original train and test datasets are transformed using the constructed features to create new feature datasets. The new train set trains the classification system, and the fitness of the chromosome g_i is determined by the classification accuracy. For regression problems, fitness is estimated by the negative mean square error between actual and predicted values.

Chromosome transformation using genetic operators: Genetic operators, crossover, and mutation are applied to form the subsequent generation of chromosomes. In the crossover process, a certain number $(n = (1 - R_s) * N_g)$ of new chromosomes are generated, replacing those with the lowest fitness in the current generation. This process involves cutting and exchanging sub-chromosomes between pairs of randomly selected parents using tournament selection. For mutation, each element in a chromosome has a chance to be changed randomly based on the mutation rate Rm.

Termination check: The process either terminates if the maximum number of generations is reached or if the best chromosome reaches a predetermined threshold of fitness (classification accuracy). Otherwise, the feature construction process restarts from the chromosome transformation using genetic operators step.

The algorithm of this approach is given in Algorithm 3.

Algorithm 3 FC2GEN algorithm

```
Ensure: split $X$ to $N_f$ parts
  while $i \leq N_f$ do (1)
      $x_i$
      For each $x_i$ construct $FT_i$ grammar
  end while
Ensure: $(x_i, t_i), i \in 1, \ldots, M$ pairs of patterns
Ensure: $N_G$
Ensure: $N_C$
Ensure: $P_S$
Ensure: $P_M$
Ensure: $N_f$
Ensure: Initialize the chromosomes in the range [0, 255].
Ensure: $iter = 1$
  while $i \leq N_g$ do
      Create a set of $N_f$ fopr the corresponding $g_i$ using (1)
      Transform original to new train data $(x_{gi,j}, t_j), j = 1, \ldots, M$
      Apply Learning C and calculate fitness $f_i$
      $$f_i = \sum_j = 1^M (C(x_{gi,j}) - t_j)^2$$
      The selection process involves categorizing chromosomes based on their fitness.
  The best-performing $(1 - P_S) \times N_C$ chromosomes, determined by their high fitness
  levels, are directly carried over to the next generation unchanged. Meanwhile, the lower-
  ranked portion of chromosomes will be replaced by new ones generated through the
  crossover procedure.
      Apply the crossover procedure. During this process, $P_S \times N_C$ chromosomes will be
  created. Two distinct chromosomes (parents) are chosen from the existing population
  using tournament selection for each pair of produced offsprings. Initially, a subset of
  $K > 1$ chromosomes is randomly selected, from which the one with the best fitness value
  is designated as a parent. Then, for each pair of parents $(z, w)$, two new offsprings, $\tilde{z}$ and
  $\tilde{w}$, are generated via one-point crossover.
      For every element of each chromosome, select a random number $r \in [0, 1]$ and alter
  the corresponding chromosome if $r \leq P_M$
  end while
  $iter = iter + 1$
  $T = (x_i, y_i), i = 1, \ldots, K$ the original test set
  Get best chromosome $g^*$ of the feature construction step
  Construct $N_F$ features for $g^*$ using (1)
  Transform $T$ into $T' = (x^*_{g,i}, t_i), i = 1, \ldots, K)$ using the previously constructed features.
  Apply a learning model such as RBF or a neural network and obtain the test error.
```

4. Performance Evaluation

Here, we provide the reader with the necessary information regarding the undertaken tests using a number of different methods as well as some results showing the efficiency of our approaches.

4.1. Simulation Setup and Metrics

The utilized dataset, referenced in [34], is purposefully curated for Software-Defined Networking (SDN) applications and is generated using the Mininet emulator. Its main objective is to facilitate the classification of network traffic through machine learning and deep learning algorithms.

The project initiates by configuring ten distinct network topologies within the Mininet environment, connecting switches with a single Ryu controller. The network simulation includes benign TCP, UDP, and ICMP traffic, alongside various types of malicious traffic, such as TCP Syn attacks, UDP Flood attacks, and ICMP attacks.

Characterized by 23 features, the dataset includes extracted attributes from switches (e.g., switch-id, packet_count, byte_count, duration_sec, and duration_nsec) and calculated parameters. The extracted information comprises the source IP, destination IP, port numbers, tx_bytes, rx_bytes, and the dt field indicating the date and time. The calculated features encompass the packet per flow, byte per flow, packet rate, packet_ins messages, total flow entries, tx_kbps, rx_kbps, and port bandwidth.

The dataset's final column acts as the class label, distinguishing between benign (labeled as 0) and malicious (labeled as 1) network traffic. The network simulation spans 250 min, resulting in a dataset comprising 104,345 rows of data.

The metric that was used in this work is the average class error, which essentially shows the percentage of a classification error using a number of algorithms. In this way, we can be certain regarding the performance efficiency of the simple grammatical evolution algorithms as opposed to other simple algorithms.

4.2. Experimental Results

Here, we use the aforementioned dataset in order to show the efficiency of our approach. We performed experiments, initially, using three methods, namely the Bayes, K-Nearest Neighbors (KNNs) [35–37], and Random Forest [38–40] methods, as opposed to GenClass. These algorithms are common in classification and they have been chosen to show the efficiency of the GenClass algorithm. Our results show that GenClass is superior to the other methods since it exhibits 6.58% error as opposed to the Bayes method error of 32.59%, KNN method error of 18.45%, and the Random Forest error of 30.70%. The results can be seen in Figure 2.

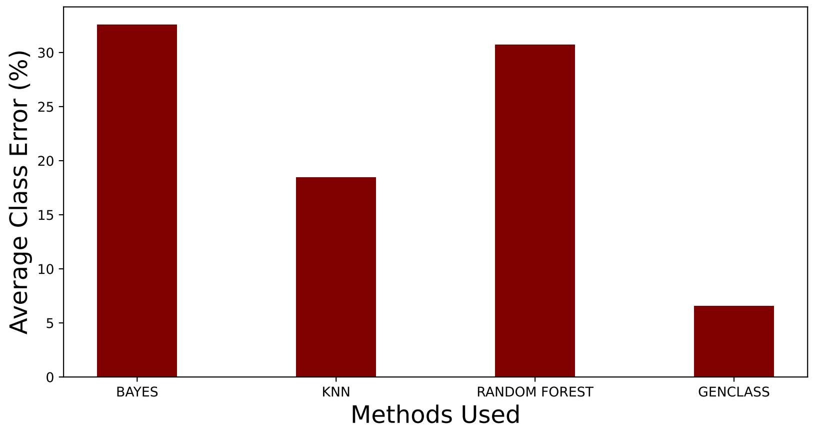

Figure 2. Error of methods in %.

Thereafter, we performed experiments using two other methods with grammatical evolution, namely the Neural Network Constructor (NNC) [32] and FC2GEN [33]. The results can be seen in Figure 3. As we can see, the GenClass method is better, exhibiting less class error comparing to the two competitors. In particular, GenClass exhibits 6.58% error, while NNC and FC2GEN exhibit average class error values of 12.51% and 15.86%, respectively. However, the results that the NNC and FC2GEN exhibited are satisfactory in terms of performance. In summary, we see that grammatical evolution provides good results.

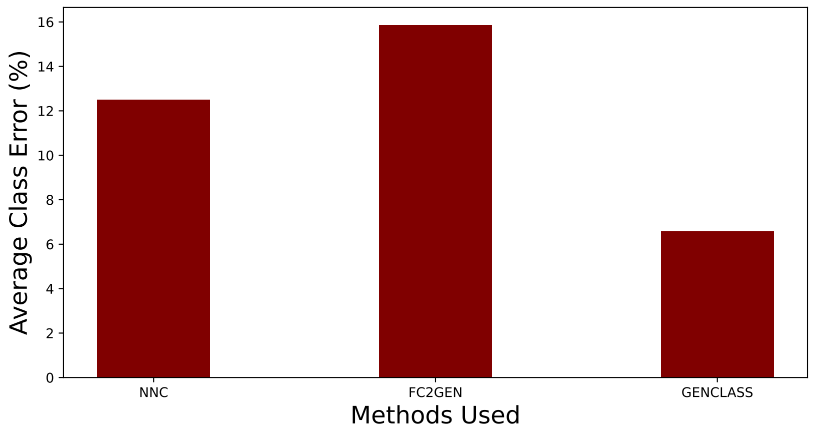

Figure 3. Error of grammatical evolution methods in %.

5. Conclusions

This paper delves into Software-Defined Networking (SDN) research, exploring a dataset encompassing a spectrum of network traffic patterns, inclusive of both benign and malicious data. This dataset encompasses 23 unique features tailored for classification purposes. The study employed grammatical evolution as its classification methodology, utilizing evolutionary algorithms to decipher intricate data patterns. This research concluded by presenting findings that assess the efficacy of the grammatical evolution-based classification approach. These results yielded valuable insights into the method's performance, contributing significantly to comprehending SDN security and the evolving landscape of network traffic classification within this dynamic field. We showed that the proposed algorithm surpasses both other classification methods and competitors from the grammatical evolution background.

For future work, we will address the work suggested in [41] in order to compare it and show how software agents could play a role in solving the problem we are addressing. Moreover, we will explore the idea of [42] to check the solution presented there in the form of a security pattern.

Author Contributions: Methodology, E.D.S.; Software, I.T.; Validation, C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been financed by the European Union: Next Generation EU through the program Greece 2.0 National Recovery and Resilience Plan, under the call RESEARCH–CREATE–INNOVATE, project name "iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques" (project code: TAEDK-06195).

Data Availability Statement: The data is publicly available.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jammal, M.; Singh, T.; Shami, A.; Asal, R.; Li, Y. Software defined networking: State of the art and research challenges. *Comput. Netw.* **2014**, *72*, 74–98. [CrossRef]
2. Xia, W.; Wen, Y.; Foh, C.H.; Niyato, D.; Xie, H. A survey on software-defined networking. *IEEE Commun. Surv. Tutor.* **2014**, *17*, 27–51. [CrossRef]
3. Chica, J.C.C.; Imbachi, J.C.; Vega, J.F.B. Security in SDN: A comprehensive survey. *J. Netw. Comput. Appl.* **2020**, *159*, 102595. [CrossRef]
4. Kanagavelu, R.; Aung, K.M.M. A survey on sdn based security in internet of things. In *Advances in Information and Communication Networks, Proceedings of the 2018 Future of Information and Communication Conference (FICC), Singapore, 5–6 April 2018*; Springer: Cham, Switzerland, 2019; Volume 2, pp. 563–577.
5. Mirkovic, J.; Reiher, P. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Comput. Commun. Rev.* **2004**, *34*, 39–53. [CrossRef]

6. Galeano-Brajones, J.; Carmona-Murillo, J.; Valenzuela-Valdés, J.F.; Luna-Valero, F. Detection and mitigation of DoS and DDoS attacks in IoT-based stateful SDN: An experimental approach. *Sensors* **2020**, *20*, 816. [CrossRef]
7. Shah, Z.; Ullah, I.; Li, H.; Levula, A.; Khurshid, K. Blockchain based solutions to mitigate distributed denial of service (DDoS) attacks in the Internet of Things (IoT): A survey. *Sensors* **2022**, *22*, 1094. [CrossRef] [PubMed]
8. Adedeji, K.B.; Abu-Mahfouz, A.M.; Kurien, A.M. DDoS Attack and Detection Methods in Internet-Enabled Networks: Concept, Research Perspectives, and Challenges. *J. Sens. Actuator Netw.* **2023**, *12*, 51. [CrossRef]
9. Cui, Y.; Qian, Q.; Guo, C.; Shen, G.; Tian, Y.; Xing, H.; Yan, L. Towards DDoS detection mechanisms in software-defined networking. *J. Netw. Comput. Appl.* **2021**, *190*, 103156. [CrossRef]
10. Santos, R.; Souza, D.; Santo, W.; Ribeiro, A.; Moreno, E. Machine learning algorithms to detect DDoS attacks in SDN. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5402. [CrossRef]
11. Singh, M.P.; Bhandari, A. New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges. *Comput. Commun.* **2020**, *154*, 509–527. [CrossRef]
12. Bawany, N.Z.; Shamsi, J.A.; Salah, K. DDoS attack detection and mitigation using SDN: Methods, practices, and solutions. *Arab. J. Sci. Eng.* **2017**, *42*, 425–441. [CrossRef]
13. Javanmardi, S.; Shojafar, M.; Mohammadi, R.; Persico, V.; Pescapè, A. S-FoS: A secure workflow scheduling approach for performance optimization in SDN-based IoT-Fog networks. *J. Inf. Secur. Appl.* **2023**, *72*, 103404. [CrossRef]
14. Javanmardi, S.; Shojafar, M.; Mohammadi, R.; Nazari, A.; Persico, V.; Pescapè, A. FUPE: A security driven task scheduling approach for SDN-based IoT–Fog networks. *J. Inf. Secur. Appl.* **2021**, *60*, 102853. [CrossRef]
15. Wang, S.; Gomez, K.; Sithamparanathan, K.; Asghar, M.R.; Russello, G.; Zanna, P. Mitigating ddos attacks in sdn-based iot networks leveraging secure control and data plane algorithm. *Appl. Sci.* **2021**, *11*, 929. [CrossRef]
16. Farris, I.; Taleb, T.; Khettab, Y.; Song, J. A survey on emerging SDN and NFV security mechanisms for IoT systems. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 812–837. [CrossRef]
17. O'Neill, M.; Ryan, C. Grammatical evolution. *IEEE Trans. Evol. Comput.* **2001**, *5*, 349–358. [CrossRef]
18. O'Neill, M.; Brabazon, A.; Ryan, C.; Collins, J. Evolving market index trading rules using grammatical evolution. In Proceedings of the Applications of Evolutionary Computing: EvoWorkshops 2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM, Como, Italy, 18–20 April 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 343–352.
19. Yang, L.; Zhao, H. DDoS attack identification and defense using SDN based on machine learning method. In Proceedings of the 2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN), Yichang, China, 16–18 October 2018; pp. 174–178.
20. Rahman, O.; Quraishi, M.A.G.; Lung, C.H. DDoS attacks detection and mitigation in SDN using machine learning. In Proceedings of the 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 8–13 July 2019; Volume 2642, pp. 184–189.
21. Sahoo, K.S.; Iqbal, A.; Maiti, P.; Sahoo, B. A machine learning approach for predicting DDoS traffic in software defined networks. In Proceedings of the 2018 International Conference on Information Technology (ICIT), Bhubaneswar, India, 20–22 December 2018; pp. 199–203.
22. Mohammed, S.S.; Hussain, R.; Senko, O.; Bimaganbetov, B.; Lee, J.; Hussain, F.; Kerrache, C.A.; Barka, E.; Bhuiyan, M.Z.A. A new machine learning-based collaborative DDoS mitigation mechanism in software-defined network. In Proceedings of the 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Limassol, Cyprus, 15–17 October 2018; pp. 1–8.
23. Alhijawi, B.; Almajali, S.; Elgala, H.; Salameh, H.B.; Ayyash, M. A survey on DoS/DDoS mitigation techniques in SDNs: Classification, comparison, solutions, testing tools and datasets. *Comput. Electr. Eng.* **2022**, *99*, 107706. [CrossRef]
24. Singh, J.; Behal, S. Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions. *Comput. Sci. Rev.* **2020**, *37*, 100279. [CrossRef]
25. Mousavi, S.M.; St-Hilaire, M. Early detection of DDoS attacks against SDN controllers. In Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC), Anaheim, CA, USA, 16–19 February 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 77–81.
26. Banitalebi Dehkordi, A.; Soltanaghaei, M.; Boroujeni, F.Z. The DDoS attacks detection through machine learning and statistical methods in SDN. *J. Supercomput.* **2021**, *77*, 2383–2415. [CrossRef]
27. Ali, T.E.; Chong, Y.W.; Manickam, S. Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review. *Appl. Sci.* **2023**, *13*, 3183. [CrossRef]
28. Cui, J.; Wang, M.; Luo, Y.; Zhong, H. DDoS detection and defense mechanism based on cognitive-inspired computing in SDN. *Future Gener. Comput. Syst.* **2019**, *97*, 275–283. [CrossRef]
29. Tonkal, Ö.; Polat, H.; Başaran, E.; Cömert, Z.; Kocaoğlu, R. Machine learning approach equipped with neighbourhood component analysis for DDoS attack detection in software-defined networking. *Electronics* **2021**, *10*, 1227. [CrossRef]
30. Tsoulos, I.G. Creating classification rules using grammatical evolution. *Int. J. Comput. Intell. Stud.* **2020**, *9*, 161–171.
31. Anastasopoulos, N.; Tsoulos, I.G.; Tzallas, A. GenClass: A parallel tool for data classification based on Grammatical Evolution. *SoftwareX* **2021**, *16*, 100830. [CrossRef]
32. Tsoulos, I.; Gavrilis, D.; Glavas, E. Neural network construction and training using grammatical evolution. *Neurocomputing* **2008**, *72*, 269–277. [CrossRef]

33. Gavrilis, D.; Tsoulos, I.G.; Dermatas, E. Selecting and constructing features using grammatical evolution. *Pattern Recognit. Lett.* **2008**, *29*, 1358–1365. [CrossRef]
34. Ahuja, N.; Singal, G.; Mukhopadhyay, D. DDOS attack SDN Dataset. *J. Netw. Comput. Appl.* **2020**. [CrossRef]
35. Ramadhan, I.; Sukarno, P.; Nugroho, M.A. Comparative analysis of K-nearest neighbor and decision tree in detecting distributed denial of service. In Proceedings of the 2020 8th International Conference on Information and Communication Technology (ICoICT), Yogyakarta, Indonesia, 24–26 June 2020; Volume 1–4.
36. Kachavimath, A.V.; Nazare, S.V.; Akki, S.S. Distributed denial of service attack detection using naïve bayes and k-nearest neighbor for network forensics. In Proceedings of the 2020 2nd International conference on innovative mechanisms for industry applications (ICIMIA), Bangalore, India, 5–7 March 2020; pp. 711–717.
37. Dong, S.; Sarem, M. DDoS attack detection method based on improved KNN with the degree of DDoS attack in software-defined networks. *IEEE Access* **2019**, *8*, 5039–5048. [CrossRef]
38. Idhammad, M.; Afdel, K.; Belouch, M. Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest. *Secur. Commun. Netw.* **2018**, *2018*, 1263123. [CrossRef]
39. Chen, Y.; Hou, J.; Li, Q.; Long, H. DDoS attack detection based on random forest. In Proceedings of the 2020 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, China, 18–20 December 2020; pp. 328–334.
40. Najar, A.A.; Manohar Naik, S. DDoS attack detection using MLP and Random Forest Algorithms. *Int. J. Inf. Technol.* **2022**, *14*, 2317–2327. [CrossRef]
41. López, J.; Maña, A.; Muñoz, A. A secure and auto-configurable environment for mobile agents in ubiquitous computing scenarios. In Proceedings of the International Conference on Ubiquitous Intelligence and Computing, Wuhan, China, 3–6 September 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 977–987.
42. Sánchez-Cid, F.; Mana, A.; Spanoudakis, G.; Kloukinas, C.; Serrano, D.; Munoz, A. Representation of security and dependability solutions. In *Security and Dependability for Ambient Intelligence*; Springer: Boston, MA, USA, 2009; pp. 69–95.