Article

Not peer-reviewed version

DDoS classification to SDN using Grammatical Evolution

Evangelos Spyrou , Ioannis Tsoulos * , Chrysostomos Stylios

Posted Date: 16 November 2023

doi: 10.20944/preprints202311.1015.v1

Keywords: SDN; DDoS; genetic algorithm; Grammatical Evolution; packet classification

Article

DDoS Classification to SDN Using Grammatical Evolution

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

[1] Department of Informatics and Telecommunications, University of Ioannina
[2] Industrial Systems Institute, Athena Research Center
* Correspondence:itsoulos@uoi.gr

Abstract: Software-Defined Networking (SDN) is a network implementation paradigm of great importance, as it has a profound impact on the pace of technological progress. While SDN doesn't directly address the technical complexities of routing, congestion control, traffic engineering, security, mobility, reliability, or real-time communication, it paves the way for innovative solutions to emerge for these and similar challenges. Security is of utmost importance in SDN with Distributed Denial of Service (DDoS) being an attack which creates large scale problems. DDoS creates malicious traffic that resembles normal traffic in order to create service problems. As such, mechanisms that distinguish between benign and malicious traffic is essential, since this is the first step to mitigate the problem of DDoS. In this paper, we take a dataset onboard which exhibits benign and malicious traffic in SDN. There are 23 features that are used for classification purposes. Here we utilise classification procedure based on three methods based Grammatical Evolution applied on the aforementioned data. We provide results that show the efficiency of our approach and show that all three methods exhibits satisfactory results.

Keywords: SDN; DDoS; genetic algorithm; Grammatical Evolution; packet classification

1. Introduction

Recent advancements in the Information and Communication Technology (ICT) domain, including mobile technologies, multimedia, cloud computing, and big data, have given rise to an increasing demand for more convenient internet access, expanded user bandwidth, and more agile service management. Software-Defined Networking (SDN) is heralded as a promising solution to meet these evolving needs. A survey on SDN is given in [1], whereby all the important aspects of SDN are identified and presented.

SDN stands as an evolving architectural approach distinguished by its dynamic, easily manageable, cost-effective, and adaptable characteristics. These qualities position SDN as exceptionally well-matched for the requirements of contemporary applications, which often demand high bandwidth and exhibit dynamic behavior. In this architectural paradigm, network control and forwarding functions are separated, enabling the direct programmability of network control and the abstraction of the underlying infrastructure to serve applications and network services.

In recent years, the scientific community has shown a growing interest in delving into the domain of SDN security in order to bridge the gap towards widespread SDN adoption. This research domain encompasses a dual focus: one side seeks to leverage SDN features to bolster security, while the other strives to establish a secure architecture for SDN systems. As we read in [2], an overview of security threats that pose risks to SDN and enumerates various attacks that exploit vulnerabilities and misconfigurations in the constituent elements of SDN are provided. Moreover, a discussion is also provided highlighting the duality between using SDN for security enhancement and securing SDN itself.

Distributed denial-of-service (DDoS) attacks pose a significant and growing threat to the Internet. Attackers consistently adapt their tactics to evade security systems, prompting researchers to continually refine their approaches to counter new attack vectors. Consequently, the DDoS

 doi:10.20944/preprints202311.1015.v1

landscape has become increasingly intricate, reaching a point where it's challenging to achieve a holistic understanding of the situation. On one hand, this complexity hinders a clear comprehension of the DDoS phenomenon, as the multitude of known attack types creates the impression of a vast and intricate problem space. On the other hand, the diverse strategies employed by existing defense systems further complicate matters, making it difficult to identify their commonalities, differences, and evaluate their effectiveness [3]. DDoS and SDN are discussed in previous works in the literature ans we can see in [4,5].

In the course of our research, we delve into the intricate domain of Software-Defined Networking (SDN) and work with a dataset that encompasses a diverse range of network traffic, encompassing both benign and malicious patterns. This dataset stands as a source of information, boasting a total of 23 distinct features that serve as the foundation for our classification attempts.

As we work on this SDN dataset, we leverage classification methodologies rooted in Grammatical Evolution. This innovative approach offers a unique perspective on the task of classifying network traffic, harnessing the power of evolutionary algorithms to make sense of the complex patterns and interactions within the data.

Our research endeavors culminate in the presentation of outcomes that offer valuable insights into the efficacy of our chosen approach. These results not only shed light on the performance of our Grammatical Evolution-based classification methods but also provide valuable contributions to the broader understanding of SDN security and the classification of network traffic in this dynamic and ever-evolving field.

2. Software Defined Networking Background

The Open Networking Foundation (ONF) [6] stands as a non-profit consortium with a clear commitment to driving forward, establishing standards, and fostering the advancement of Software-Defined Networking (SDN) for commercial purposes. This organization has been instrumental in presenting a widely embraced and easily understandable definition of SDN. In essence, SDN embodies a burgeoning network architecture that effectively separates network control from data forwarding, allowing for direct programmability [7].

According to this definition, SDN is defined by two core attributes: the segregation of control and data planes, and the ability to program the control plane. However, it's crucial to acknowledge that these features are not entirely novel in the realm of network architecture, as elaborated upon in the subsequent discussion.

Initially, numerous efforts have been made to render networks programmable. Active networking, for instance, endeavors to manage networks using software methods [8]. Additionally, routers have been manipulated via software to enhance the programmability of network devices [9,10]. These network devices' functionalities can be adjusted by either installing new routing software or modifying existing software configurations.

The notion of separating control and data planes has gained significant traction over the last decade. In their work, [11], authors introduced the Routing Control Platform (RCP) as a replacement for the Border Gateway Protocol (BGP) in inter-domain routing. This centralized routing control aimed to streamline the complex process of fully distributed path computation. Additionally, the Internet Engineering Task Force (IETF) introduced the Forwarding and Control Element Separation (ForCES) framework, which effectively segregates control and packet forwarding elements within a ForCES Network [12,13].

Moreover, [14,15] presented a 4D approach, offering a comprehensive redesign of the entire network architecture through four distinct planes: "decision," "dissemination," "discovery," and "data," organized from top to bottom. This conceptual framework aimed to address various functionalities and responsibilities within the network structure.

Furthermore, the Path Computation Element (PCE) architecture was introduced to autonomously compute label-switched paths independently of the actual packet forwarding process in MPLS and

 doi:10.20944/preprints202311.1015.v1

Generalized GMPLS networks [16]. These advancements collectively represent substantial progress in delineating and refining the separation of control and data planes within network architectures.

The works highlighted in [17,18] introduced a paradigm shift by enhancing fundamental flow-based Ethernet switches with a centralized controller responsible for managing the admission and routing of data flows. This innovative approach placed significant emphasis on separating the data and control planes. Notably, this method has gained notable traction within the commercial networking equipment domain. For example, Cisco's ASR 1000 series routers and Nexus 7000 series switches stand as prime illustrations. These systems have intelligently decoupled and modularized the control plane from the data plane. This architectural decision enables the simultaneous operation of an active control plane instance alongside a standby counterpart, ensuring robust fault tolerance and seamless transitions during software upgrades. This approach marks a considerable advancement in ensuring network stability, reliability, and adaptability within the industry.

In the context of SDN, its distinctiveness lies in its ability to offer programmability by distinctly separating the control and data planes. SDN fundamentally revolutionizes how we program network devices, providing a user-friendly alternative to the intricate nature of active networking. Moreover, SDN champions the segregation of control and data planes within the network's architectural framework. This separation allows for the independent management of network control on the control plane without disrupting the data flow. Consequently, it enables the extraction of network intelligence from switching devices, relocating it to controllers. Simultaneously, this setup allows external software to manage switching devices without requiring embedded intelligence.

The decoupling of the control plane from the data plane not only fosters a more simplified, programmable environment but also delivers enhanced flexibility for external software to shape the network's behavior. This transformation marks a pivotal shift in how networks are managed and controlled, streamlining the process and opening avenues for a more adaptable and responsive network infrastructure.

3. DDoS Overview

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 [3]. 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, 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 are orchestrated through interconnected networks of internet-enabled devices, encompassing both traditional computers and a diverse array of devices, such as Internet of Things (IoT) devices. These devices are compromised by malware, effectively placing them under the remote control of an attacker. Together, these compromised devices are commonly referred to as 'bots' or 'zombies,' forming what is known as a 'botnet.'

Once a botnet is established, the attacker gains the ability to coordinate an attack by issuing remote instructions to each bot within the network. When a victim's server or network becomes the target of the botnet, each bot begins sending numerous requests to the victim's IP address. This influx of traffic has the potential to overwhelm the server or network, leading to a disruption of regular services, ultimately causing a denial of service.

The establishment of a botnet enables the attacker to wield substantial power over the coordinated assault, leveraging the sheer volume of traffic generated by the network of compromised devices to overwhelm the target, thereby impeding its regular operations.

 doi:10.20944/preprints202311.1015.v1

Mitigating Distributed Denial-of-Service (DDoS) attacks presents an inherent challenge due to the fact that each bot involved mimics a legitimate internet device, making it arduous to distinguish the malicious attack traffic from regular, legitimate data.

The primary indication of a DDoS attack typically manifests as a sudden decrease in website or service speed, and in extreme cases, complete unavailability. However, because various factors, including legitimate traffic surges, can cause similar performance issues, a comprehensive investigation is usually necessary. To discern specific indicators of a DDoS attack, the use of traffic analysis tools becomes imperative.

It's important to note that there exist additional, more specialized telltale signs of a DDoS attack, which can vary based on the specific type of attack being executed. These attacks encompass application-layer, protocol-based, and volumetric assaults that infiltrate and disrupt the functioning of a network. Recognizing these varied attack types is crucial in implementing effective defense strategies against the distinct modes of DDoS attacks.

4. Related Work

In the paper referenced in [19], 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 (ANN), 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 [20], 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 of 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 [21], 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 [22], 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 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 [23], 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 swift restoration of regular communication. Their findings emphasize the system's

 doi:10.20944/preprints202311.1015.v1

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 [24] utilize the Neighbourhood 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.

5. Dataset Description

The dataset that has been employed [25], was specifically curated for application within the realm of Software-Defined Networking (SDN) and is generated using the Mininet emulator. Its primary purpose is to facilitate the classification of network traffic through machine learning and deep learning algorithms.

The project commences by configuring ten distinctive network topologies within the Mininet environment, interconnecting switches with a single Ryu controller. The network simulation encompasses benign TCP, UDP, and ICMP traffic, as well as various types of malicious traffic, including TCP Syn attacks, UDP Flood attacks, and ICMP attacks.

This dataset is characterized by 23 features, comprising a mix of extracted attributes from the switches and calculated parameters. Extracted attributes include details such as Switch-id, Packet_count, byte_count, duration_sec, and duration_nsec (expressed in nanoseconds). It also encompasses information like Source IP, Destination IP, Port numbers, tx_bytes (representing the bytes transmitted from the switch port), and rx_bytes (indicating the bytes received on the switch port). Additionally, the dt field captures date and time, converted into numerical values, with monitoring intervals set at 30 seconds.

Calculated features include Packet per flow, Byte per flow (representing the byte count within a single flow), Packet Rate (denoting the number of packets sent per second, calculated by dividing the packet count by the monitoring interval), as well as metrics like the number of Packet_ins messages, total flow entries in the switch, tx_kbps, rx_kbps, indicating data transfer and receiving rates, and Port Bandwidth, sum of tx_kbps and rx_kbps.

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

6. Proposed Methodologies

Here we provide the reader with three proposed methods using Grammatical Evolution (GE) to perform the necessary tasks that are required.

6.1. GenClass

The GenClass method [26] as presented in various related publications [27–29], has demonstrated 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 [30]. Grammatical evolution in this context relies on the following key components:

Primarily, the grammar of the target language, expressed in 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 $A \to a$ or $A \to aB$, where A, B belong to N, and a belongs to T. Secondarily, the corresponding fitness function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 November 2023 doi:10.20944/preprints202311.1015.v1

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. Before that we introduce the reader with the nomenclature of the parameters used.

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 chromosome is initialised randomly in r_m.
Ensure: $iter = 1$
while $i \leq N_g$ **do do** (1)
Create C_i for g_i
Calculate $f_i = \sum_{i=1}^{M} (C(x_i) - t_i)^2$
Utilize the selection process. In this phase, the chromosomes are categorized based on their fitness. The top-performing $(1 - P_S) \times N_C$ chromosomes remain unaltered and are passed on to the subsequent generation in the population. The remaining chromosomes will be substituted by new chromosomes generated during the crossover.
Implement the crossover method. In this procedure, $P_S \times N_C$ chromosomes are generated. Initially, for each pair of newly produced offspring, two unique chromosomes (parents) are chosen from the existing population using tournament selection: A subset of more than one $(K > 1)$ randomly selected chromosomes is formed, and the chromosome with the most superior fitness value is chosen as the parent. For every parent pair (z, w), two new offspring, $\tilde{z}, \tilde{w}$, are created using the one-point crossover method.
Execute the mutation process. For every component of every chromosome, pick a random number, r, from the range between $[0, 1]$, and modify the respective 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

6.2. Neural Network Construction (NNC)

Tsoulos et al. [31] 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 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.

 doi:10.20944/preprints202311.1015.v1

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 $iter\%Li = 0$ **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.

6.3. FC2GEN

The section summarizes the work in [32,33], which constitutes the third method that we utilised 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.

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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 November 2023 doi:10.20944/preprints202311.1015.v1

Feature Construction from Chromosome Parts: The chromosome is divided into N_f equal parts, each part $(g_i, i = 1, ..., N_f)$ is used to construct a feature. Features $(f_i, i = 1, ..., N_f)$ are created from each part g_i through a mapping process. This construction process can be executed in parallel.

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 step Chromosome Transformation using Genetic Operators.

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: $(x_i, t_i), i \in 1, ..., M$ 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 $(x_{g_{i,j}}, t_j), j = 1, ..., M$
    Apply Learning C and calculate fitness f_i

$$f_i = \sum_j = 1^M (C(x_{g_{i,j}}) - t_j)^2$$

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, ∼z and ∼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
$iter = iter + 1$
$T = (x_i, y_i), i = 1, ..., K$ the original test set
Get best chromosome g^* of the feature construction step
Construct N_F features for g^* using (1)
Transform T into $T' = (x_{g_i^*}, t_i), i = 1, ..., K)$ using the previously constructed features.
Apply a learning model such as RBF or a neural network to and obtain the test error.
=0

7. Results

Here, we use the aforementioned dataset in order to show the efficiency of our approach. We performed experiments, initially, using 4 methods, namely Bayes, K-Nearest Neighbours (KNN) and Random Forest. Our results show that GenClass is superior to the other methods since it exhibits 6.58

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 November 2023 doi:10.20944/preprints202311.1015.v1

% error as opposed to Bayes 32.59, KNN with 18.45 %, and Random Forest with 30.70 %. The results can be seen in Figure 1.

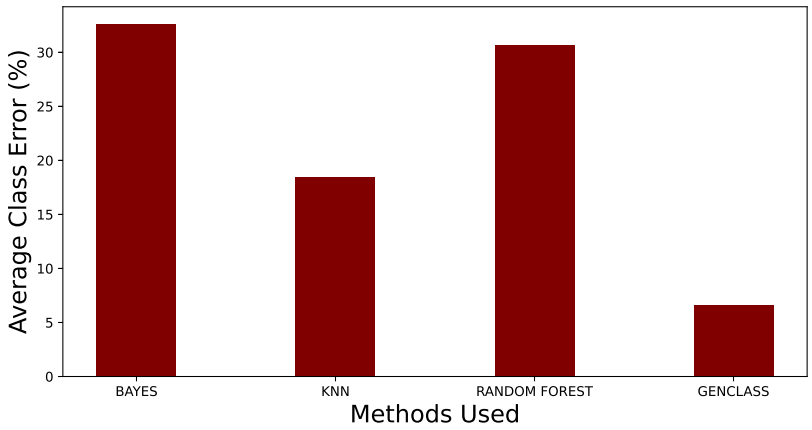

Figure 1. Error of Methods in %.

Thereafter we performed experiments using two other method with Grammatical Evolution, namely the Neural Network Constructor (NNC) [31,34] and the FC2GEN [32]. The results can be seen in Figure 2. As we can see the GenClass method is better, exhibiting less class error comparing to the two competitors. In particular, GenClass exhibits 6.58 % while NNC and FC2GEN exhibit average class error 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.

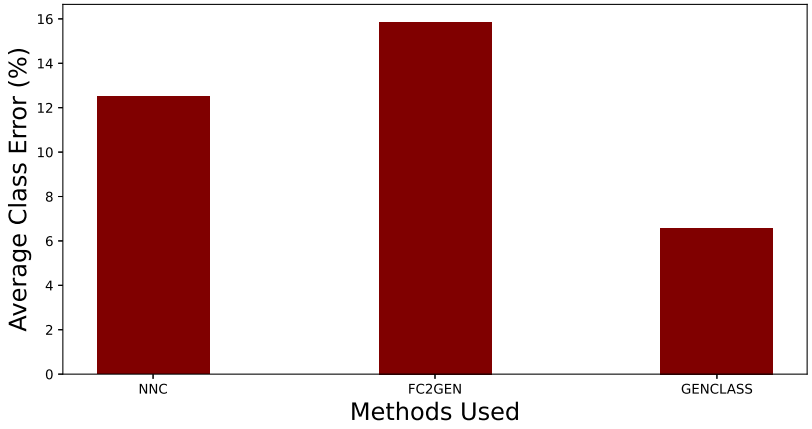

Figure 2. Error of Grammatical Evolution Methods in %.

8. Conclusions

The 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 employs Grammatical Evolution as its classification methodology, utilizing evolutionary algorithms to decipher intricate data patterns.

The research concludes by presenting findings that assess the efficacy of the Grammatical Evolution-based classification approach. These results yield 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 show that the proposed algorithm surpasses both other classification methods as well as competitors from the Grammatical Evolution background. For future work, we leave the utilisation of parallel and federated approach to edge DDoS attacks.

Acknowledgments: 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)

References

1. Xia, W.; Wen, Y.; Foh, C.H.; Niyato, D.; Xie, H. A survey on software-defined networking. *IEEE Communications Surveys & Tutorials* **2014**, *17*, 27–51.
2. Chica, J.C.C.; Imbachi, J.C.; Vega, J.F.B. Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications* **2020**, *159*, 102595.
3. Mirkovic, J.; Reiher, P. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review* **2004**, *34*, 39–53.
4. Cui, Y.; Qian, Q.; Guo, C.; Shen, G.; Tian, Y.; Xing, H.; Yan, L. Towards DDoS detection mechanisms in software-defined networking. *Journal of Network and Computer Applications* **2021**, *190*, 103156.
5. Santos, R.; Souza, D.; Santo, W.; Ribeiro, A.; Moreno, E. Machine learning algorithms to detect DDoS attacks in SDN. *Concurrency and Computation: Practice and Experience* **2020**, *32*, e5402.
6. (ONF), O.N.F. [Online]. Available: https://www. opennetworking.org/.
7. Software-defined networking: The new norm for networks, Palo Alto, CA, USA, White Paper **Apr. 2012.**
8. Alexander, D.S.; Arbaugh, W.A.; Hicks, M.W.; Kakkar, P.; Keromytis, A.D.; Moore, J.T.; Gunter, C.A.; Nettles, S.M.; Smith, J.M. The SwitchWare active network architecture. *IEEE network* **1998**, *12*, 29–36.
9. Kohler, E.; Morris, R.; Chen, B.; Jannotti, J.; Kaashoek, M.F. The Click modular router. *ACM Transactions on Computer Systems (TOCS)* **2000**, *18*, 263–297.
10. Handley, M.; Hodson, O.; Kohler, E. XORP: An open platform for network research. *ACM SIGCOMM computer communication review* **2003**, *33*, 53–57.
11. Feamster, N.; Balakrishnan, H.; Rexford, J.; Shaikh, A.; Van Der Merwe, J. The case for separating routing from routers. Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture, 2004, pp. 5–12.
12. Lakshman, T.; Nandagopal, T.; Ramjee, R.; Sabnani, K.; Woo, T. The softrouter architecture. Proc. ACM SIGCOMM Workshop on Hot Topics in Networking, 2004, Vol. 2004.
13. Wang, W.; Dong, L.; Zhuge, B.; Gao, M.; Jia, F.; Jin, R.; Yu, J.; Wu, X. Design and implementation of an open programmable router compliant to IETF ForCES specifications. Sixth International Conference on Networking (ICN'07). IEEE, 2007, pp. 82–82.
14. Rexford, J.; Greenberg, A.; Hjalmtysson, G.; Maltz, D.A.; Myers, A.; Xie, G.; Zhan, J.; Zhang, H. Network-wide decision making: Toward a wafer-thin control plane. Proc. HotNets, 2004, pp. 59–64.
15. Greenberg, A.; Hjalmtysson, G.; Maltz, D.A.; Myers, A.; Rexford, J.; Xie, G.; Yan, H.; Zhan, J.; Zhang, H. A clean slate 4D approach to network control and management. *ACM SIGCOMM Computer Communication Review* **2005**, *35*, 41–54.
16. A. Farrel, J.P.V.; Ash, J. A Path Computation Element (PCE)-Based Architecture **2006**.
17. Casado, M.; Garfinkel, T.; Akella, A.; Freedman, M.J.; Boneh, D.; McKeown, N.; Shenker, S. SANE: A Protection Architecture for Enterprise Networks. USENIX Security Symposium, 2006, Vol. 49, p. 50.
18. Casado, M.; Freedman, M.J.; Pettit, J.; Luo, J.; McKeown, N.; Shenker, S. Ethane: Taking control of the enterprise. *ACM SIGCOMM computer communication review* **2007**, *37*, 1–12.
19. Singh, J.; Behal, S. Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions. *Computer Science Review* **2020**, *37*, 100279.
20. Mousavi, S.M.; St-Hilaire, M. Early detection of DDoS attacks against SDN controllers. 2015 international conference on computing, networking and communications (ICNC). IEEE, 2015, pp. 77–81.

21. Banitalebi Dehkordi, A.; Soltanaghaei, M.; Boroujeni, F.Z. The DDoS attacks detection through machine learning and statistical methods in SDN. *The Journal of Supercomputing* **2021**, *77*, 2383–2415.
22. Ali, T.E.; Chong, Y.W.; Manickam, S. Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review. *Applied Sciences* **2023**, *13*, 3183.
23. Cui, J.; Wang, M.; Luo, Y.; Zhong, H. DDoS detection and defense mechanism based on cognitive-inspired computing in SDN. *Future generation computer systems* **2019**, *97*, 275–283.
24. 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.
25. Ahuja, N.; Singal, G.; Mukhopadhyay, D. DDOS attack SDN Dataset. *Journal of Network and Computer Applications* **2020**.
26. Tsoulos, I.G. Creating classification rules using grammatical evolution. *International Journal of Computational Intelligence Studies* **2020**, *9*, 161–171.
27. Christou, V.; Tsoulos, I.; Arjmand, A.; Dimopoulos, D.; Varvarousis, D.; Tzallas, A.T.; Gogos, C.; Tsipouras, M.G.; Glavas, E.; Ploumis, A.; others. Grammatical Evolution-Based Feature Extraction for Hemiplegia Type Detection. *Signals* **2022**, *3*, 737–751.
28. Spyrou, E.D.; Stylios, C.; Tsoulos, I. Classification of CO Environmental Parameter for Air Pollution Monitoring with Grammatical Evolution. *Algorithms* **2023**, *16*, 300.
29. Arjmand, A.; Christou, V.; Tsoulos, I.G.; Tsipouras, M.G.; Tzallas, A.T.; Gogos, C.; Glavas, E.; Giannakeas, N. An evolutionary algorithm-based optimization method for the classification and quantification of steatosis prevalence in liver biopsy images. *Array* **2021**, *11*, 100078.
30. Anastasopoulos, N.; Tsoulos, I.G.; Tzallas, A. GenClass: A parallel tool for data classification based on Grammatical Evolution. *SoftwareX* **2021**, *16*, 100830.
31. Tsoulos, I.; Gavrilis, D.; Glavas, E. Neural network construction and training using grammatical evolution. *Neurocomputing* **2008**, *72*, 269–277.
32. Gavrilis, D.; Tsoulos, I.G.; Dermatas, E. Selecting and constructing features using grammatical evolution. *Pattern Recognition Letters* **2008**, *29*, 1358–1365.
33. Tsoulos, I.G. QFC: A Parallel Software Tool for Feature Construction, Based on grammatical evolution. *Algorithms* **2022**, *15*, 295.
34. Tsoulos, I.G.; Tzallas, A.; Tsalikakis, D. NNC: A tool based on Grammatical Evolution for data classification and differential equation solving. *SoftwareX* **2019**, *10*, 100297.