Vivek Kumar, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Using Model Explanation, Deep Learning Can Be Used To Detect And Classify Botnet Traffic

Mridula Singh[1], Shilpy Sharma[2], Sagar Chaudhary[3]

[1,2,3]Assistant Professor, Department of CSE, Quantum University, Roorkee, India

Abstract- Malicious attempts known as distributed denial-of-service attacks because target services to be unavailable to legitimate users by sending many service requests that exceed the processing capacity of the services. Detection of botnet traffic is therefore critical to maintaining the availability and quality of the services while identification of the type of botnet attacks helps system administrators quickly figure out which part of the computer and network system are under attacks. The focus of existing research is on rule-based detection, which establishes rules in the network firewall to deny suspicious traffic that matches the rules. The emergence of machine learning and deep learning (ML/DL) has led to the development of preliminary works to learn botnet traffic behavior and perform detection. It is possible to enhance the performance of existing ML/DL models, but their decision-making and prediction are not transparent, which makes it hard for users to interpret and trust the results. In this work, we develop a novel deep learning model for botnet detection and classification, which has the ability to explain the model's decisions. Convolutional neural networks' latent representation of traffic feature allows us to detect if a traffic record is generated by a bot and then identify the type of bot responsible for the record. The predictions of the developed deep learning model are interpreted using an existing explainable framework. Extensive experiments are conducted with both realistic network traffic and synthetic traffic generated by the IXIA Breaking Point System. Various performance metrics are used to compare the developed model with existing models. The test results indicate that the developed model surpasses the current machine learning classifiers by up to 15% on all performance metrics, while providing a straightforward explanation of the model decision.

Keywords: Network Security, Botnet Detection and Classification, Deep Learning, Explainable AI.

I. INTRODUCTION

Significant security threats to computer users have been posed by the rapid growth of botnets. The number of victims of botnets and smart devices has significantly increased. Untraceable and exceptional coordinated mechanisms are used by hackers to attack a computer or a network for their illegal activities. In order to accomplish this objective, a group of hosts in different geographical locations, which are compromised and controlled by a malicious individual or organization, launch the attack simultaneously. The complexity of the Internet makes it difficult to trace back to the origin, posing serious threats to legitimate Internet activities. Information leakage, click fraud, and Email spam can be the first step of a serious problem as using these vulnerabilities a botnet can infect smart devices or computers and then launches more serious attacks in networks like (distributed) denial

of service (DDoS) attacks [1]. Botnets, which are emerging threats, are infecting thousands of hosts worldwide. The majority of these hosts use Microsoft Windows operating systems (OS 2) and (OS 3) The term 'bots' is frequently used to describe software applications running as an automated task over the Internet. A specific group of bots under a command and control.

C2 or C&C) server is able to form a self-propagating, self- organizing, and autonomous network, named as botnet [2]. The C&C server will remotely control bots using different kinds of communication protocols to install worms, Tro- jan horses, or backdoors on the systems to compromise them. The strategy evolution between botnet's masters and botnet defenders are ongoing to win the battle. The bot- net's strength lies in the massive number of bots, which increases the severity of attacks and its master's ability to hide the bots from being detected by security systems.

In the current Internet malware epidemic, botnets are one of the main threats as bots are able to cooperate towards a common malicious purpose and can spread over thousands of computers at a very high speed [4]. An example of popular botnets is Mirai botnet that spreads through Trojans and exploited Internet-of-Things (IoT) devices such as web cameras, closed circuit television cameras (CCTV), and other devices with low-security measures and involved 100,000 IoT devices [2]. Detection and classification of botnet traffic become critical for network security to protect the systems and stored data.

The research related to detection of different botnet

Families has been a trending topic over the last decade [5] and [6]. With the emergence of machine learning and deep learning (ML/DL), several works have proposed to apply machine learning and deep learning to network security problems [7], [8] and botnet detection [9], [10], [11], [12]. The continuous changing of network measurement statistic and botnet behaviour characteristic that will cause existing rule- based approaches to fail. On the other hand, convolutional neural networks (CNN) consistently show improved performance in identifying and recognising complex pattern in computer vision and other fields [13]. Generally, 2D convo- lutional neural networks are used as images are 2D data. But network traffic in raw format is 1D data. So, in this paper, we develop 1D Convolutional Neural Network (1DCNN) classifier to leverage the capability of representation of latent data feature of convolutional neural networks and use it for identification of types of botnets and normal traffic. We develop a tool for traffic feature extraction to provide an enriched set of features that are useful for the convolution layers to learn the best latent representation of benign and botnet traffic.

While ML/DL have demonstrated their capabilities in performing complex data analytic tasks, their prediction or decision mechanism is still a black-box for researchers and practitioners. The existing works in botnet detection and classification only reports their prediction result and do not provide any explanation of their outcome. Those models are mostly handled by the security analysts (who mostly are not machine leaning experts) to whom the constructed models seem like a black-box without any transparency and clear evidence to explain why a given traffic record is identified as a generated traffic from a particular type of botnet attack or as a normal traffic. It is generally difficult for an analyst to manually analyze the highly complex black-box model like deep neural network with millions of parameters and infer the rationale behind the decision making [14].

As a result, the analysts cannot determine whether they could trust the decisions made by the model. To mitigate this problem, vari- ous explanation approaches have been proposed to interpret predictions generated from the complex machine learning- based models [15], [16], [17], [18]. These methods explain a classification model using additive feature attribution framework, i.e., they identify the input feature set which the classification gave most importance while producing its prediction. By examining these most input features, domain expert could easily verify whether the

classifier following right logic according intrinsic property of the domain.

In this paper, we adopt one such popular framework
named SHAP [16] to the network security problem to ex- plain the prediction of the proposed botnet detection and classification model to earn the trust of its end users, i.e., security analysts. The output of the explanation framework (i.e., the SHAP value of the features) allows us to under- stand not only the importance of features on the nature of traffic samples but also the impact direction of their im- portance, i.e., whether a feature negatively or positively contributes to the class decision of the traffic samples. The contributions of this paper are as follows.

1. We develop a novel botnet detection and classifi- cation model using 1DCNN, which is able to learn complex pattern of botnet traffic.
2. We adopt an explanation framework to provide explanations about the outcomes of botnet detection and classification model to earn the trust of end users.
3. We carry out extensive experiments to demonstrate the performance of the detection and classification model, and assess the explanation quality of the explanation framework. We use three datasets (two realistic traffic datasets and a synthetic dataset gen- erated by IXIA Appliance1) to generalize the effec- tiveness of the 1DCNN model and the explanation framework.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the details of the proposed framework. Section 4 presents the experimental setup and the analysis of the experimental results. We conclude our paper in Section 5.

II. RELATED WORK

Botnet attacks could cause tremendous damage to organiza- tions and governments, especially those who provide critical services to the public. Besides, the episodes could signifi- cantly disrupt online activities such as financial transactions, online shopping, chatting, etc. The first botnet was appeared in 2001 to send spam emails. Since then, many have studied botnet behavior and proposed different methods to detect the botnet. These botnet detection methods are commonly applied at the host level or network level. At the host-based level, the method detects botnets by monitoring the host system behavior for any botnet-related suspicious activities. On the other hand, the network-based detection method captures and analyzes network traffic data for identifying hosts infected with botnets. Furthermore, as machine learning and deep learning have shown good detection performance in various other domains, many have proposed botnet detection based on the two learning approaches as discussed.

In host-based detection, several works have been proposed as follows. The framework"BotSwat" [19] was pro- posed to detect infected machines by botnets by monitoring the launched host processes using external parameters. Another framework, "MineSweeper" [9] detects botnets by analyzing all hidden behaviors of the botnets. Cui et al. [10] proposed a botnet detection algorithm that uses outbound connections of the processes to the internet, i.e., monitoring the inputs from mouse, keyboard, and other sources. Law et al. [11] proposed a botnet detection method by capturing memory and network events from the machines.

Subsequently, the authors performed correlating them to identify suspicious devices. A clustering method [12] is proposed to detect botnets by clustering user behavior executions especially targeting anomalies. Finally, Oulehla et al. [20] proposed a mobile botnet detection based on the deep learn- ing approach. The method can detect mobile botnets by clas- sifying mobile traffic data. All of the above discussed host- based detection methods for botnets have some limitations. For example, many advanced botnets could use evasive techniques to escape detection in [10], [19]. The solution [9] is hard to operate in a fully automated manner and partly still needs to involve humans, and some scalability issues in handling the large dataset, etc. in [12].

Host-based detection does not have the global visibility of the network, thus failing to detect botnet traffic where bots are distributed. Network-based detection overcomes this issue by collecting traffic at the network perimeter or

1. IXIA Appliance

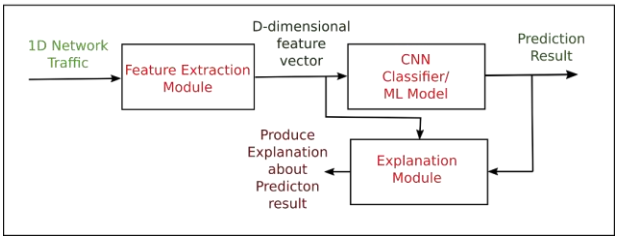

Fig. 1. The overall architecture of the Botnet detection and classification framework.

at different points of the network. The network-based bot- net detection is categorized into signature-based, anomaly- based, the domain name server (DNS), and machine learn-

Feature Extraction Module

In this paper, we define various features that can be used by various machine learning (ML) algorithms. It is worth mentioning that by carefully selecting features from raw network packets, and from aggregated flows, algorithms can learn to detect and classify various type of botnet traffic in a network. Depending on aggregation criteria, different features can be extracted for real-time detection and classi- fication. We first provide the definitions of traffic flows and sessions in terms of Raw traffic or Packets and then present detailed description of the aggregation technique used here.

3.1.1 Packet

A raw traffic stream is consist of a set of packets P = {p1, . . . , p|P |}. Each packet pi is defined by a tu- ple {sip , dip , sporti, dporti, proto , pktsize }, where sip ,

ing/deep learning-based. Signature-based techniques [21], i i

i i i

[22] find botnets by searching and matching the exist- ing known botnet signatures in the network traffic. The anomaly-based methods [7], [23], [24], [25] use anomaly fea- tures extracted from the network traffic to identify and de- tect botnets. The DNS-based approaches [26], [27], [28] could identify botnets by monitoring DNS traffic which the botnet uses DNS to communicate a command-and-control (C & C) server and detecting any anomalies in DNS queries. For the machine learning/deep learning-based approaches [29], [30] use additional network traffic and other related features to train various botnet detection models. [29], [30] extracted the above statistical features to train botnet DDoS detection models using supervised learning methods such as SVM, Na¨ıve Bayes, etc. Similarly, [31], [32], [33], [34], [35] pro- posed DDoS botnet detection models using different neural network architectures such as Convolutional Neural Net- work (CNN) [33], [34], [35], Gated Recurrent Unit Neural Network (GRU) [31] and Recurrent Neural Network/Long Short-Term Memory Neural Network (RNN/LSTM) [32], [34]. All these botnet detection and classification algorithms do not provide any explanation about their prediction. We provide explanation along with the prediction results to earn the trust of domain expert.

3 PROPOSED FRAMEWORK FOR BOTNET DETEC-TION AND CLASSIFICATION WITH EXPLANATION

The proposed botnet detection and classification framework has three distinct modules, namely Feature extraction mod- ule, Botnet detector and classifier module, i.e., 1DCNN model and Explanation module. The schematic diagram of the proposed framework is shown in Fig. 1. The network traffic is 1-dimensional data. Here, the traffic stream first enters into the Feature extraction module. This module extracts the 43 feature values, which include basic features in packet headers and statistical features after performance traffic aggregation into flows and sessions. These features are then enters into 1DCNN model for detection and clas- sification. The explanation module produces impact factor of these features using SHAP values [16]. We provide the details of the component modules in the following subsec- tions.

dipi, sporti, dporti, protoi and pktsizei are source IP ad- dress, destination IP address, source Port, destination Port, protocol used by the packet and size of the packet in bytes, respectively [36].

3.1.2 Flow

A traffic flow is a sequence of packets, localized in time, identified by the 5-tuple <source IP address, Source port, Destination IP address, destination port, protocol> such that the inter-arrival between two consecutive packets is less than a threshold value, α seconds. If two packets , pi and pj share the same 5-tuple information, but have an inter arrival time greater than the threshold value α, then they are aggregated into two different flows. In this paper, we used α = 5, 10, 15 seconds for flow generation.

3.1.3 Session

As defined above, flows have an inherent notion of di- rection. This creates situations where an application-level requests or commands to be part of one flow, while the response packets are part of a separate flow. To address this situation, we define a traffic session. In a session, the next level of aggregation happens by combining corresponding requests and their reply flows together. Most NetFlow or IPFIX implementations are unidirectional, meaning TCP connections between two hosts results in two flows (i.e., A to B and B to A) that can be stitched back into one bidirectional flow.

To capture more statistical features that could be use- ful for various botnet attack detection, two or more uni- directional flows could be aggregated into a session that have some common basis to put together. For instance, if a malicious host carries out a scan attack on different victims, the source IP, source port and protocol should be the same for all the generated flows. These flows could then be aggregated into one session. We define a temporal criterion for aggregating flows to create a session. If the inter-arrival time between two consecutive flows is greater than a given threshold value β then these two flows will be part of two separate sessions. The shorter the value of β enables the detection and classification framework to flag any botnet traffic quickly but in turn it may loose some important characteristic patterns in the derived statistical features that are vital for ML algorithms.

3.2 Feature Aggregation

In this paper, we extract basic traffic features and produce 4 types of features to feed in the 1DCNN model.

1) Aggregated features: These features are computed as a sum, total count, max or min value of a simple feature over a flow or a session using uni- or bi- directional flows.
2) Temporal features: These are duration parameters (in seconds) of a flow or a session and their sum, mean and standard deviation.
3) Statistical features: These are statistical measures such as mean and standard deviation of aggregated features, over a flow or session of component byte and packets numbers.
4) Connection-context features: These features are useful in detecting coordinated attacks by several machines in a compromised network and can be used to recognize multi-flow attacks. “Flow wa- termarking” identifies sets of flows that have the same/co-related patterns [37] .

We tabulated the description of all the 199 features that we extracted and derived from basic traffic features for The Stratosphere IPS Project [38] and Synthetic data on Appendix in Table 8 and 9. We encourage interested reader to look into them. We extracted traffic features with different values of β (01 and 03 minutes) while varying the value of α (05 and 10 seconds). It is worth mentioning that the shorter the value of β, the faster the detection system will classify Botnets but it can also miss out the botnet traffic as the extracted features may not be sufficient to differentiate with benign traffic.

3.3 Botnet Detection and Classification Module

The application of the convolutional framework in deep neural network has been a great success for computer vision problem. We would like to test this framework for network traffic data. Unlike image data, the network traffic is a 1D data. So, we use 1D convolutional framework. We empiri- cally evaluated various deep neural network architecture on DS4 dataset of Table 2 and tabulated their performance in Table 1. We used batch-normalization, dropout probability is 0.8, number of epoch as 1500, activation function as leaky- relu

with leakage factor as 0.1 and Class-Balanced Focal Loss (section 3.3.3) as loss function with parameters β1 = 0.9 and γ = 2.1. We keep all of these parameter same for all the architecture. Of Table 1, the first column represents architec- ture number, second column states number of CNN layers used in architecture, third column tabulates the details of CNN layers (filter size x channel number x number of filters and ';' separates layer description), forth column mentions

Fig. 2. The F1-score of various deep network architecture.

• If we increase the fully connected (FC) layer more than 3, the performance of the network does not improve at all. Sometime the performance of the network degrades.

Keeping all of these observation in mind, in this paper we use Architecture 12 to detect and classify Botnet traffic from network traffic data. The network traffic data first went to feature extraction module. This module produces samples/records for input to the 1DCNN model. Each sam- ple has 199 features values extracted from the traffic data. The 1DCNN model has 5 convolution (CNN) layers, 3 fully connected (FC) layers, and one softmax layer. The details schematic of the 1DCNN classification model is shown in Figure 3. The right side of a convolutional layer has details of the filter size used in it. We used batch-normalization at end of convolutional layers to improve the generalization capability of the network. The size of FC layers mentioned in the box which represents that FC layer. We used Max- pooling layers between CNN convolutional layers. We used leakyRelu activation for all of the Neurons. We tested the following 3 loss functions in our framework. The rationale behind using them is they are well capable of handling class imbalanced data. We minimized them using Adam optimiser.

3.3.1 Macro-averaged F 1-score

The mathematical formulation of this loss function is as follows

$$loss = 1 - F1; \quad F1 = \frac{1}{M}\sum; \quad F1_i = 2 * Pi * i, \qquad (1)$$

number of Fully Connected (FC)layers used, third column =

$$c=1 = \frac{R\,Pi + Ri}{}$$

tabulates the details of FC layers (input neuron size x output neuron size; ':' separates layer description) and finally last shows the f1-score in % on test set which mutually exclusive set from training set. The F1-score of various deep learning architecture is shown in Fig. 2. From Table 1 and Fig 2, it is shown that

• If we increase the convolution layer the performance of the network improves up to layer number 5 after than its performance saturates.

where Pi and Ri are the Precision and Recall of the class i and M is the number of class. Precision represent the ratio of positive identifications that is actually correct. Recall repre- sent the ratio of actual positives that is identified correctly. Here 'positive' indicates samples from the intended class. The advantage of F 1-score as a loss function is that small value of either precision or recall will result in lower overall score. Thus it helps balance the two metrics. This increases its sensitivity of detecting minority class [39].

TABLE 1
Performance of various 1DCNN architectures

Architure #	# of CNN Layers	Description of CNN layers	# of FC Layers	Description of FC layers	Test set F1-score
1	2	9x1x64;5x1x128	3	6400x512:512x256:256x12	88.21
2	3	13x1x32;9x1x64;5x1x128	3	3200x256:256x12	90.35
3	3	13x1x32;9x1x64;5x1x128	3	3200x200:200x200:200x12	90.45
4	3	13x1x32;9x1x64;5x1x128	3	3200x400:400x400:400x12	90.30
5	3	13x1x32;9x1x64;5x1x128	3	3200x512:512x256:256x12	90.60
6	3	13x1x32;9x1x64;5x1x128	4	3200x1024:1024X512:512x256:256x12	90.00
7	4	17x1x16;13x1x32;9x1x64;5x1x128	3	3200x512:512x256:256x12	91.14
8	4	17x1x32;13x1x32;9x1x64;5x1x128	3	3200x512:512x256:256x12	91.27
9	5	21x1x32;17x1x64;13x1x128;9x1x192;5x1x256	3	3328x512:512x256:256x12	91.83
10	5	17x1x32;13x1x64;9x1x128;5x1x192;3x1x256	3	3328x512:512x256:256x12	91.57
11	5	23x1x32;19x1x64;15x1x128;11x1x192;7x1x256	2	3328x256:256x12	92.03
**12	5	23x1x32;19x1x64;15x1x128;11x1x192;7x1x256	3	3328x512:512x256:256x12	92.10
13	5	23x1x32;19x1x64;15x1x128;11x1x192;7x1x256	4	3328x1024:1024X512:512x256:256x12	91.09
14	5	27x1x32;21x1x64;15x1x128;11x1x192;5x1x256	3	3328x512:512x256:256x12	91.85
15	6	27x1x32;23x1x64;19x1x128;15x1x160;11x1x192;7x1x256	3	1792x512:512x256:256x12	91.82
16	6	23x1x32;19x1x64;15x1x128;11x1x160;7x1x192;5x1x256	3	1792x512:512x256:256x12	92.03
17	7	31x1x32;27x1x64;23x1x128;19x1x160:15x1x192;11x1x224; 7x1x256	3	3328x512:512x256:256x12	92.24
18	7	31x1x32;27x1x64;23x1x128;19x1x160:15x1x192;11x1x224; 7x1x256	2	3328x256:256x12	91.85
19	7	29x1x32;25x1x64;21x1x128;17x1x160:13x1x192;9x1x224; 5x1x256	3	3328x512:512x256:256x12	91.93

3.3.2 Focal Loss

The cross entropy (CE) loss is popularly used as a loss function in deep learning framework. It performs reason- ably well in balanced dataset. But the large class imbalance encountered during training of dense detectors overwhelms the cross entropy loss [40]. Easily classified samples from majority classes comprise the significant part of the loss and dominate the gradient. The Focal loss is designed by reshaping the cross entropy loss function to down-weight easy examples and thus focus training on hard sample i.e. sample from minority classes [40]. The mathematical formulation is as follows

$$loss = -1 * \sum_{c=1}^{M} (1 - pc)^{\gamma} * \log(pc), \quad (2)$$

where $\gamma \geq 0$ is a focusing parameter which smoothly adjusts the rate at which easy samples are

down-weighted and pc ∈ [0, 1] is the model's estimated probability of a sample being in the class with label c

3.3.3 Class-Balanced Focal Loss

The Class-Balanced Loss is designed to address the problem of training from imbalanced data by introducing a weight- ing factor that is inversely proportional to the effective number of samples [41]. The formulation of Class-Balanced

we this loss function in our deep learning framework for performing our experiment. To prevent over-fitting of the model, we used Drop-out layers along with all FC layers expect with the output layer and added l2-regularisation along with loss function.

3.4 Explanation Module

As we used a complex hard-to-interpret machine learning models like deep neural networks for network traffic classi- fication purposes, Interpretability is crucial to earn trust of its end user. Existing work on explaining complex models can be divided into two main categories; global and local explanations depending whether the framework tries to describe the model as a whole i.e. using all of its prediction or try to identify how the different input variables/features influenced a specific prediction/output from the model. Explanation methods may further be divided into two categories: model-specific and model-agnostic (general) ex- planation methods depending on whether the explanation framework is specific to a particular type of machine learn- ing module.

In this paper, we use a popular model agnostic global ex- plainable AI technique, SHAP [16] to explain predictions of the novel malware detection and classification model. This

framework is based on the Shapely value which is a method

Focal Loss is

loss = −1 ∗

1 − β

1

Σ

∗ M(1 − p)γ ∗ log(p), (3)

originally invented for assigning "payouts" to "players" depending on their contribution towards the total "payout"

1 − β1nc c c c=1

where pc ∈ [0, 1] is the model's estimated probability of a sample being in the class with label c, γ ≥ 0 is a focusing parameter, nc is the number of sample in class c and β1 ≥ 0 determines the number of effective samples of a class and it is a data dependent parameters.

Among these 3 losses the Class-Balanced Focal Loss is performed best. So, unless it is pacifically mentioned,

and builds on concepts from cooperative game theory [42]. In the explanation setting, the features are the "players" and the prediction is the total "payout". SHAP stands for SHapley Additive exPlanations and focuses on the differ- ence between the prediction and the average prediction is perfectly distributed among the features. For instance, if the prediction to be explained is the probability of person P getting symptomatic after getting contacted with COVID- 19 virus, the sum of the Shapley values for all features (in

method which the backbone of SHAP framework [16]. This method requires less computational power to obtain a simi- lar approximation accuracy.

3.4.1 Shapley Value Computation

The exact shapley value is computed using cooperative game theory. Let, there are D players in total and let P ⊆ D be a subset consisting of |P| players. Here, D is full feature set of network traffic data. A contribution function v(P) that maps subsets of players to the real numbers, called the worth or contribution of coalition P and describes the total expected sum of "payouts" the members of P can obtain by cooperation. From this total "payout" a player k gets its fair share according to Shapely value is as follows

ϕk(v) = ϕk

Σ
=
P⊆D\{k}

|P|!(D − |P| − 1)!
(v(P∪{k})−v(P));
D!

(4)

where k = 1 . . . D. So, we can consider ϕk is a weighted mean over contribution function differences for all subsets P of players not containing player k [16].

3.4.2 Mapping to Machine Learning Scenarios

Let there be a training set: {xiyi}, i = 1, . . . , nT , a predictive model: f (x) and target vector: Y. nT is the size of training set. Our goal is explain prediction from the model f (xs) for a specific sample x = xs using Shapley. We achieve this goal by representing the total "payouts" in this case f (x =

xs) as follows
f (xs) = ϕ0 +

ΣD

k=1

sϕ ; ϕ0 = E[f (x)]; (5)

this case is the person P 's vital parameters, gene mark-ups etc) is equal to the difference between this prediction and the mean probability of a person getting symptomatic after getting contacted with COVID-19 virus, where the mean is taken over all persons having contacted with COVID-19 virus. "all persons" include all age group, gender, all types of vital parameters, etc.

The advantage of Shapley value is it has a solid theoret- ical foundation compared to other explainable AI method and due to this foundation it Local accuracy, Missingness and Consistency property. The main disadvantage of the Shapley value is that it becomes in-tractable for high dimensional data as the computational complexity grows exponentially. This has led to approximations like the Kernel SHAP

where E[f (x)] is the global average prediction and ϕs is the individual "payouts" of feature k for the prediction of a specific sample x = xs. The Shapley values explain the difference between the prediction ys = f (xs) and the global average prediction.

When we consider the contribution for a certain subset P∫ , we modify the contribution function as v(P) and com- pute it as follows

v(P) = E[f (x)|xP = xPs]. (6)

Using Eqs. 4, 5, and 6 and Kernel trick SHAP framework efficiently computed the Shapley value for individual fea- ture [16].

Fig. 3. The Botnet detection and classification aka. 1DCNN framework. i represents sample or record i and j is jth feature. X is the set of input samples or records.

4 EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION OF FRAMEWORK

4.1 Datasets

In order to evaluate the performance of the proposed frame- work, we used below datasets:

- The Stratosphere IPS Project [38] is continually ob- taining malware and normal traffic and store in PCAP files. The dataset includes 11 botnet families that generates different attacks that spread over the Internet. Using various combination of α and β for traffic sampling and feature extraction, we have created 6 datasets from the raw traffic (i.e., PCAP files). Each dataset contains traffic samples (which

TABLE 2
Class-wise number of records/samples from the Stratosphere IPS Project dataset

Software type	Software	DS 1 α = 5s β = 1m	DS 2 α = 10s β = 1m	DS 3 α = 5s β = 3m	DS 4 α = 10s β = 3m
Benign	Normal	1299434	1109930	1379736	1131939
Malware	Andromeda	216452	215234	216452	215234
	Barys	101333	83612	101333	83612
	Emotet	442501	423762	442501	204075
	Dridex	87800	84822	87800	84822
	Artemis	73069	71507	73069	71507
	Miuref	131919	123080	131919	123080
	Necurs	27025	25533	27025	25533
	Sality	724718	310009	825596	79628
	Trickbot	749324	753586	779471	724446
	Ursnif	11269	10924	11269	10924
	Wannacry	41266	40069	41266	40069

TABLE 3
Class-wise number of records/samples from Kitsune dataset

Software Type	Software Family	Kitsune
Benign	Normal	16166317
Malware	Active Wiretap	923216
	ARP MitM	1145272
	Fuzzing	432783
	OS Scan	65700
	SSDP Flood	1439604
	SSL Renegotiation	92652
	SYN DoS	7038
	Video Injection	102499
	Mirai	642516

are 1-dimensional feature vectors) to be classified as benign or malicious with specific class of botnet. The class-wise sample distribution of the datasets mentioned in Table 2. We have extracted features with different aggregation criteria discussed in Section 3.2. The name and description of 199 features are presented in Table 8 and Table 9 in Appendix A.

We refer Kitsune Surveillance Network Intrusion Datasets [43] as Kitsune dataset in our paper. This dataset consist of 9 types of botnet attacking traffic namely

1) Active Wiretap: A bridged Raspberry Pi placed be- tween all cameras and the DVR server
2) ARP MitM: ARP Man-in-the-Middle attack between a camera & DVR
3) Fuzzing: A Fuzzing Attack against DVR's web- server's cgi
4) OS Scan: NMAP OS Scan of the subnet
5) SSDP Flood: SSDP Flooding Attack against the DVR Server
6) SSL Renegotiation: A DoS attack against an SSL enabled camera
7) SYN DOS: A SYN DoS attack against a camera
8) Video Injection: A MitM video content injection attack into a camera's live stream
9) Mirai: The initial infection and propagation of the Mirai malware on a diffrent [IoT] network

The class-wise sample distribution of the Synthetic datasets mentioned in Table 3.

A synthetic dataset generated using IXIA PerfectStorm One: BreakingPoint System (BPS) by simulating enterprises, government agencies traffic, and general Internet traffic

TABLE 4
Class-wise number of records/samples from Synthetic dataset

Software Type	Software Family	Synthetic
Benign	Normal	482524
Malware	DNS Flood	416
	HTTP Flood	78375

UDP Flood 84983
UDP-PF 227
VSEQF 16208

using various protocols. BPS generates real time attack traffic that exploits and exposes security vulnerabilities on application servers, security devices, network devices and enterprise infrastructure components. The synthetic dataset has the normal IT traffic and Mirai botnet attack traffic. It contains 5 types of Mirai Botnet attack application profile including DNS Flood, HTTP Flood, UDP Flood, UDP Plain Flood (UDP-PF) and Value Source Engine Query Flood (VSEQF). This dataset has 6 classes in total including normal traffic. We have created this dataset from the synthetic Internet traffic using α = 10s and β = 1m. The class-wise sample distribution of the Synthetic datasets mentioned in Table 4. The feature aggregation scheme and description is same as Stratosphere IPS Project dataset.

4.2 Analysis of Experimental Results

Here, we compare the performance of novel Botnet detector and classifier module i.e. 1DCNN models with other well known Machine Learning (ML) models. We divide dataset into 2 mutually exclusive sets, namely training set and test set. The training set contains 80% of the sample that are in dataset. The test set contains the remaining 20%. We used decision tree (DT), Random forest (RF) and Multi-layer perception (MLP) model for comparison. The performance results on test set of 4 datasets described in Table 2 are in Table 5. We evaluated the performance of ML and DL models using 5 evaluation metrics namely Accuracy, Pre- cision Score, F1-score, Area Under the Curve of Receiver Operating Characteristic (AUC-ROC) and Area Under the Curve of Precision Recall Curve (AUC-PRC) and tabulated their scores in columns 3 to 7 respectively in Table 5. The first and second column contain dataset name and the classifier name respectively. The 1DCNN classifier model performed better in all evaluation metrics and in all datasets. There is upto 5% improvements of its performance in terms of Accuracy and AUC-PRC.

The Precision Recall Curve and AUC-ROC Curve on datasets DS1, DS2, DS3 and DS4 of various classifiers are shown in Figs. 4 to 7 respectively. After examining all the figures we found that the 1DCNN classifier performed better with respect to all other classifiers. This performance improvements more prominently visible in Precision Recall Curves (Figs 4a - 7a).

We measure the performance of ML and DL models us- ing 5 evaluation metrics on Kitsune and Synthetic data. The performance results on test set are in Table 6. The test was created using 20% of the total data. Its samples do not have any overlap with training data. The structure of this table is same as Table Table 5. We put the result of these two datasets in separate table because Kitsune data does not have same

TABLE 5
Performance comparison of 1DCNN classifier model with other ML models with four Stratosphere IPS Project datasets. All evaluation metrics are in percentage.

Data Set Name	Classifier Models	Evaluation Metrics			
		Accuracy	Precision		
		F1-score	AUC-ROC	AUC-PRC	
DS 1 α = 5s β = 1m	DT	85.00	89.96	84.01	96.90
		90.32			
	MLP	87.86	94.16	87.45	98.57
		94.17			
	RF	85.20	89.95	84.21	97.27
		90.37			
	1DCNN	92.23	98.00	92.14	99.56
		97.96			
DS 2 α = 10s β = 1m	DT	87.01	90.15	86.13	97.12
		90.54			
	MLP	88.00	93.51	87.00	98.63
		93.52			
	RF	86.54	89.99	83.55	97.88
		90.01			
	1DCNN	92.98	98.34	92.88	99.62
		98.34			

DS 3					
α = 5s					
β = 3m	DT	86.56	90.47	85.56	96.77
	90.83				
	MLP	88.54	94.73	88.24	98.60
	94.93				
	RF	85.16	90.01	83.17	97.34
	90.00				
	1DCNN	93.30	98.50	93.28	99.63
	98.44				
DS 4					
α = 10s					
β = 3m	DT	86.39	88.66	84.38	97.04
	89.06				
	MLP	87.54	94.23	87.98	98.76
	94.55				
	RF	87.46	92.36	85.56	98.58
	92.38				
	1DCNN	92.08	97.50	92.10	99.50
	97.48				

(a) Precision Recall Curve

(b) AUC-ROC Curve

Fig. 4. The Precision Recall Curve and AUC-ROC Curve on dataset DS2 of various classifiers. Legend showing the name of the classifiers is used.

feature as with Stratosphere IPS Project data and second data is not a real data i.e. we generate it in our using lab IXIA PerfectStorm One machine. The 1DCNN model performs better with respect to other machine learning models with respect to 5 evaluation metrics for these two datasets. For Synthetic data it performs nearly at 100% in all evaluation metrics. One reason could be that IXIA PerfectStorm One machine could not able to simulate all realistic network

(a) Precision Recall Curve

(b) AUC-ROC Curve

Fig. 5. The Precision Recall Curve and AUC-ROC Curve on dataset DS2 of various classifiers. Legend showing the name of the classifiers is used.

traffic scenario.

The Precision Recall Curve and AUC-ROC Curve gener- ated from Kitsnue and Synthetic datasets on various classi- fiers are shown in Fig. 8 and Fig. 9, respectively. From both figures we concluded that the 1DCNN model performed better than other classifiers. As the 1DCNN model achieved nearly 100% AUC-ROC and AUC-PRC on test set of syn-

TABLE 6

Performance comparison of 1DCNN model with other ML models with Kitsune and Synthetic datasets. All evaluation metrics are in percentage.

Data Set Name	Classifier Model	Evaluation Metrics			
		Accuracy	Precision		
		F1-score	AUC-ROC	AUC-PRC	
Kitsune	DT	87.51	77.83	87.50	87.35
	88.13				
	MLP	91.20	98.10	91.01	99.50
	98.09				
	RF	92.20	97.35	92.10	99.83
	97.43				
	1DCNN	97.90	99.85	97.72	99.90
	99.80				
Synthetic		DT	96.71	99.51	96.11
	99.52	99.49			
	MLP	95.84	99.48	95.83	99.48
	99.37				
	RF	96.95	99.39	95.97	99.51
	99.41				
	1DCNN	99.99	99.99	99.99	99.99
	99.99				

(a) Precision Recall Curve

(b) AUC-ROC Curve

Fig. 6. The Precision Recall Curve and AUC-ROC Curve on dataset DS3 of various classifiers. Legend showing the name of the classifiers is used.

thetic data, the plots of them touch 100% value on the graph.

4.3 Explanation of Model Decision

As we discussed in section 3.4, SHAP generates explana- tion of predictions of a classifier by identifying the most important features based on a features feature attribution framework and Shapley value. In simple words SHAP values tell us that the classifier mostly depends on these features for its classification task. To compute SHAP value according to Eq. 5, we need to compute E[f (x)] over a set of sample x. Due to our resource [we used graphics card (Nvidia GTX 2080, 11 GB) in our the computing system] limitation, we could only be able to use 30 samples for this set. As we would like to examine how 1DCNN classifier predicts each class individually, we are interested on those samples which are correctly classified by the classifier. We

(a) Precision Recall Curve

(b) AUC-ROC Curve

Fig. 7. The Precision Recall Curve and AUC-ROC Curve on dataset DS4 of various classifiers. Legend showing the name of the classifiers is used.

did a k-means clustering [39] with cluster number k = 30 on these correctly classified the network traffic records for each class. We use these 30 cluster mean as x to E[f (x)] for SHAP framework. Here we include results from dataset DS2. We randomly select 500 samples/records from test set for each class to produce explanations in form of SHAP value. We have shown the top 20 class-wise important features by sorting the sum of magnitudes of SHAP value over these samples for DS2 in Fig. 11 and Fig. 12. These plots are called summary plots. It is worth mentioning that the actual class and predicted class by 1DCNN model of these selected 500 samples are same. The SHAP values indicate the impacts of each feature on the model's output; in this case, predicting a particular class. The color shown in the plots represent the relative magnitude of feature value: red indicates high value whereas blue indicate low. As we are using 500 samples, we

(a) Precision Recall Curve

(b) AUC-ROC Curve

Fig. 8. The Precision Recall Curve and AUC-ROC Curve on Kitsune dataset of various classifiers. Legend showing the name of the classi- fiers is used. Clearly 1DCNN classifier outperformed all the classifiers.

have 500 SHAP values for each feature. In the plots, the straight line above zero indicate the neutral line represents the E[f (x)]. For examples, if all the SHAP values for a feature fall on this line, that feature does not have any impact on the model outcome. The SHAP values that fall right side of this line has positive impact of a specified range of a feature values and that are in the left side of these line have negative impact.Here positive impact of a feature means the specified range of values of the feature lead a classifier to classify the samples from class in concern correctly, i.e., the specified range help traffic records fall within the boundary of the class in concern. The negative impact means the vice-versa.

Some important observations using SHAP framework are as follows.

1) The high session duration values (sess dur) with session duration with 1 minute has negative im- pact for Normal class (Fig. 11a) where as the same has positive impact for botnet class data specially for Artemis (Fig. 11f), Sality (Fig. 12c), Trickbot (Fig. 12d), Ursnif (Fig. 12e) and Wannacry (Fig. 12f).

2) The high value of sport has negative impact for Normal class (Fig. 11a) where as the same has positive impact for most of the botnet classes, e.g., Andromeda (Fig. 11b), Artemis (Fig. 11f), Miuref (Fig. 12a), Wannacry (Fig. 12f), etc. One important observation is that the high value of sport has

negative impact in classification for Necurs class (Fig. 12f). This makes this unique among the other

(a) Precision Recall Curve

(b) AUC-ROC Curve

Fig. 9. The Precision Recall Curve and AUC-ROC Curve on Synthetic dataset of various classifiers. Legend showing the name of the classi- fiers is used.

botnet class.

3) The byte rate related features (e.g., sum bytes rate, max bytes rate, etc.) and packet rate related features (e.g., max tot pktsrate, min tot pktsrate, etc.) have high significant for classification among various bot- nets (Fig. 11 and Fig. 12) compare to Normal class. The Andromeda, Necurs and Sality class data have very high dependence on these type of features for correct classification.

4) The high value of dport has negative impact on Normal class data. The Dridex and Emotet botnet have high sensitivity to this feature and due to this they are unique with respect to other botnet classes. The high of this has positive impact on these class of data Fig. 11d and Fig. 11e.

5) We have extracted forward and backward direction features separately. None of them appeared in the top 20 feature list with respect to SHAP values. In our future study we may not extract them from data. This information is helpful for on line implementa- tion of this framework.

These observations highlight the fact that the models outcomes have parity with the domain knowledge. These supporting facts are helpful for earning trust from end user, i.e., security expert. This signed and graded information of relative feature information are not available with the set of features selected using RF/DT framework. They select features which will discriminate between class boundaries. But these framework fail to highlight relative effects of features on class boundary formation. We also obtained

similar summary plots on the SHAP values on the DS1, DS3, DS4 and synthetic datasets. Due to space limitation, we do not put them in this paper.

(a) Precision Recall Curve of Architecture 12

(b) AUC-ROC Curve of Architecture 12

(c) Precision Recall Curve of Architecture 17

(d) AUC-ROC Curve of Architecture 17

Fig. 10. The Precision Recall Curve and AUC-ROC Curve on dataset DS4 of various loss functions of Architecture 12 and 17. Legend shows the names of the losses used.

TABLE 7
F1-score comparison of 1DCNN model varying loss functions on dataset DS4. All scores are in percentage

Architecture Number	Class Name	Loss Name		
		F1-loss	FL	CBFL
	Normal	87.27	93.35	93.13
12	Andromeda	92.31	94.04	93.68
	Barys	98.16	99.00	98.97
	Emotet	81.43	90.97	90.87
	Dridex	78.19	97.24	96.93
	Artemis	66.78	76.64	59.92
	Miuref	74.19	84.27	84.34
	Necurs	60.13	87.98	84.29
	Sality	85.01	96.52	96.08
	Trickbot	92.63	94.32	94.16
	Ursnif	24.02	00.00	39.63
	Wannacry		82.03	96.35	95.83
	Overall	86.92	92.73	92.10
17	Normal	88.34	93.27	93.45
	Andromeda		83.67	94.12	94.17
	Barys	97.92	98.87	99.33
	Emotet	81.77	90.90	90.71
	Dridex	90.69	96.72	95.92

Artemis	70.81	76.62	59.07	
Miuref	79.38	83.62	84.98	
Necurs	78.39	87.37	84.51	
Sality	84.55	96.42	96.19	
Trickbot	92.78	94.18	94.25	
Ursnif	20.58	00.00	41.80	
Wannacry		88.23	95.84	97.02
Overall	87.50	92.61	92.24	

4.4 Ablation Study

We did an ablation study of Architecture 12 and 17 by varying three loss functions as discussed in Section 3.3 on dataset DS4. We choose these two architecture as they are best performing among the others according to Fig. 2. The class-wise and over all F1-score on test data of F1-loss, Focal Loss (FL), and Class Balanced Focal Loss (CBFL) functions are tabulated in columns 3, 4 and 5 of Table 7, respectively. The first two columns of this table contain architecture number that is coming from Table 1 and class name of the data, respectively.

Though in terms of over all F1-score FL loss is best, the F1-score with respect to Ursnif class is zero for both architectures. Ursnif is class with smallest samples. So, FL loss fails to detect single sample from this class of botnet. From cybersequrity point of view, this is not acceptable. In case of F1-loss, the F1-score is low for the overall as well as other classes with respect to other losses. Only the architectures trained with CBFL able to detect samples from all the class with relatively high F1-score. The Precision Recall Curve and AUC-ROC Curve on datasets DS4 of with respect to these three loss functions are shown in Figure 10 for both architectures. The performance of FL and CBFL are almost similar and are fur better than the same with F1- loss. We study the performances of DS1, DS2, DS3 and Kitsune datasets and get similar results. After this ablation study, we decided to use CBFL loss function to train our architectures.

5 CONCLUSION

Though botnet detection and classification is a well studied problem, we are motivated by lack of efforts in gaining the trust of end-users through interpretation of model decision. In this paper, we not only designed a novel botnet detection and classification framework but also provided evidences and explanation to trust its outcome. We have used this framework on four datasets created from publicly available pCAP files from Stratosphere IPS Project and one Kitsune dataset by varying different parameters of feature extraction method to detect and classify various botnet traffics that are present in them. We have designed a novel 1DCNN model which outperformed other state-of-the-art ML models in terms of classification performance. We also provided the explanation of the 1DCNN model's outcome through SHAP framework. Their results demonstrated that its predictions have parity with the domain knowledge as well as its outcome is highly trustworthy. The fact is import for gaining trust for its end users.

APPENDIX

FEATURE DESCRIPTION

The detailed description of features used in The Strato- sphere IPS and Synthetic data are presented in Table 8 and Table 9.

REFERENCES

[1] K. Ono, I. Kawaishi, and T. Kamon, "Trend of botnet activities," in 2007 41st Annual IEEE International Carnahan Conference on Security Technology, 2007, pp. 243–249.

[2] W. N. H. Ibrahim, S. Anuar, A. Selamat, O. Krejcar,
R. Gonza´lez Crespo, E. Herrera-Viedma, and H. Fujita, "Mul- tilayer framework for botnet detection using machine learning algorithms," IEEE Access, vol. 9, pp. 48 753–48 768, 2021.

[3] Wikipedia. Botnet. [Online]. Available: https://en.wikipedia.org/ wiki/Botnet

[4] J. Govil, "Examining the criminology of bot zoo," in 2007 6th Inter- national Conference on Information, Communications Signal Processing, 2007, pp. 1–6.

[5] M. Eslahi, R. Salleh, and N. B. Anuar, "Bots and botnets: An overview of characteristics, detection and challenges," in 2012 IEEE International Conference on Control System, Computing and Engineering, 2012, pp. 349–354.

[6] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, and J. Zhang, "Botnet: Classification, attacks, detection, tracing, and preventive measures," EURASIP J. Wirel. Commun. Netw., vol. 2009, Feb. 2009. [Online]. Available: https://doi.org/10.1155/2009/692654

[7] T. Truong-Huu, N. Dheenadhayalan, P. Pratim Kundu, V. Ram- nath, J. Liao, S. G. Teo, and S. Praveen Kadiyala, "An Empiri- cal Study on Unsupervised Network Anomaly Detection Using Generative Adversarial Networks," in 1st Security and Privacy on Artificial Intelligent Workshop (SPAI'20), Taipei, Taiwan, Oct. 2020.

[8] T.-D. Pham, T.-L. Ho, T. Truong-Huu, T.-D. Cao, and H.-L. Truong, "MAppGraph: Mobile-App Classification on Encrypted Network Traffic using Deep Graph Convolution Neural Networks," in Annual Computer Security Applications Conference (ACSAC 2021), Virtual Conference, December 2021.

[9] D. Brumley, C. Hartwig, Z. Liang, J. Newsome, D. Song, and H. Yin, "Automatically identifying trigger-based behavior in mal- ware," in Botnet Detection. Springer, 2008, pp. 65–88.

[10] W. Cui, R. H. Katz, and W.-t. Tan, "Binder: An extrusion-based break-in detector for personal computers," in USENIX Annual Technical Conference, General Track, 2005, pp. 363–366.

[11] F. Y. Law, K.-P. Chow, P. K. Lai, and K. Hayson, "A host-based approach to botnet investigation?" in International Conference on Digital Forensics and Cyber Crime. Springer, 2009, pp. 161–170.

[12] H. Lamba, T. J. Glazier, J. Ca´mara, B. Schmerl, D. Garlan, and J. Pfeffer, "Model-based cluster analysis for identifying suspicious activity sequences in software," in Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics, 2017, pp. 17–22.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, p. 84–90, may 2017. [Online]. Available: https://doi.org/10.1145/3065386

[14] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," ACM Comput. Surv., vol. 51, no. 5, Aug. 2018. [Online]. Available: https://doi.org/10.1145/3236009

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1135–1144. [Online]. Available: https://doi.org/10.1145/2939672.2939778

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[17] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1675–1684. [Online]. Available: https://doi.org/10.1145/2939672. 2939874

[18] J. Kazemitabar, A. Amini, A. Bloniarz, and A. S. Talwalkar, "Variable importance using decision trees," in Advances in Neural Information Processing Systems, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/5737c6ec2e0716f3d8a7a5c4e0de0d9a-Paper.pdf

[19] E. Stinson and J. C. Mitchell, "Characterizing bots: Remote control behavior," in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2007, pp. 89–108.

[20] M. Oulehla, Z. K. Oplatkova´, and D. Malanik, "Detection of mobile botnets using neural networks," in 2016 Future Technologies Conference (FTC). IEEE, 2016, pp. 1324–1326.
[21] SnortIDS. [Online]. Available: http://www.snort.org
[22] M. Roesch et al., "Snort: Lightweight intrusion detection for net- works." in Lisa, vol. 99, no. 1, 1999, pp. 229–238.
[23] M. Yamada, M. Morinaga, Y. Unno, S. Torii, and M. Takenaka, "Rat-based malicious activities detection on enterprise internal networks," in 2015 10th International Conference for Internet Tech- nology and Secured Transactions (ICITST). IEEE, 2015, pp. 321–325.
[24] D. Jiang and K. Omote, "An approach to detect remote access trojan in the early stage of communication," in 2015 IEEE 29th international conference on advanced information networking and appli- cations. IEEE, 2015, pp. 706–713.
[25] J. Liao, S. G. Teo, P. Pratim Kundu, and T. Truong-Huu, "ENAD: An Ensemble Framework for Unsupervised Network Anomaly Detection," in Proc. IEEE CSR 2021, Virtual Conference, July 2021.
[26] A. Ramachandran, N. Feamster, and D. Dagon, "Detecting botnet membership with dnsbl counterintelligence," in Botnet Detection. Springer, 2008, pp. 131–142.
[27] H. R. Zeidanloo and A. B. A. Manaf, "Botnet detection by monitoring similar communication patterns," arXiv preprint arXiv:1004.1232, 2010.
[28] D. Mahjoub, "Monitoring a fast flux botnet using recursive and passive dns: A case study," in 2013 APWG eCrime Researchers Summit. IEEE, 2013, pp. 1–9.
[29] D. Zammit, "A machine learning based approach for intrusion prevention using honeypot interaction patterns as training data," University of Malta, pp. 1–55, 2016.
[30] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in 2018 IEEE Security and Privacy Workshops (SPW). IEEE, 2018, pp. 29–35.
[31] X. Yuan, C. Li, and X. Li, "Deepdefense: identifying ddos attack via deep learning," in 2017 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, 2017, pp. 1–8.
[32] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Bre- itenbacher, and Y. Elovici, "N-baiot-network-based detection of iot botnet attacks using deep autoencoders," IEEE Pervasive Comput- ing, vol. 17, no. 3, pp. 12–22, 2018.
[33] S.-C. Chen, Y.-R. Chen, and W.-G. Tzeng, "Effective botnet detec- tion through neural networks on convolutional features," in 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, 2018, pp. 372–378.
[34] B. Nugraha, A. Nambiar, and T. Bauschert, "Performance evalua- tion of botnet detection using deep learning techniques," in 2020 11th International Conference on Network of the Future (NoF). IEEE, 2020, pp. 141–149.
[35] S. Hosseini, A. E. Nezhad, and H. Seilani, "Botnet detection using negative selection algorithm, convolution neural network and classification methods," Evolving Systems, pp. 1–15, 2021.
[36] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for represen- tation learning," in 2017 International Conference on Information Networking (ICOIN), 2017, pp. 712–717.
[37] N. Moustafa, "Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic," Ph.D. dissertation, 10 2017.
[38] S. R. Laboratory. Labeled dataset with botnet, normal and background traffic. [Online]. Available: https://www.stratosphereips.org/
[39] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics). Berlin, Heidelberg: Springer-Verlag, 2006.
[40] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dolla´r, "Focal loss for dense object detection," in 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2999–3007.
[41] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019,

pp. 9260–9269.
[42] L. S. Shapley, A Value for N-Person Games. Santa Monica, CA: RAND Corporation, 1953.
[43] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in Network and Distributed Systems Security (NDSS) Symposium, 2018, appears in Network and Distributed Systems Security Sym- posium (NDSS) 2018; null ; Conference date: 18-02-2018 Through 21-02-2018