Educational Administration: Theory and Practice
2024, 30(4), 7655-7667
ISSN: 2148-2403
https://kuey.net/

Research Article

DCRNN: Deep Convolution Reinforcement Neural Network-Based Cyber-Attacks Detection In Iot Data

Dr. Sivakumar. T[1*], Ms. Chaithra Varshini V[2]

[1*]Associate Professor and HOD In BCA Karnataka College Management and Science, Bengaluru, Karnataka
[2]Assistant Professor, Department of Computer Applications Karnataka College Management and Science, Bengaluru, Karnataka

Citation: Dr. Sivakumar. T, Ms. Chaithra Varshini V, DCRNN: Deep Convolution Reinforcement Neural Network-Based Cyber-Attacks Detection In Iot Data, *Educational Administration: Theory And Practice, 30(4), 7655-7667*
Doi: 10.53555/kuey.v30i4.2621

ARTICLE INFO **ABSTRACT**

Honeyed framework is a system environment for defending legitimate network resources against attack. The Honeyed framework fosters resource-stealing behavior by encouraging attackers to use it. This is a procedure for detecting an attack using an attack detection procedure. To recognize denial of service (DoS) threats, we employ the Honeyed framework system in this research. Primary security devices to prevent your network by enabling the identification of attacks in the face of network attacks are NIDS (Network Intrusion Detection Systems). We propose a system that exposes an attack and verifies a defense mechanism against the same attack in this paper. For the new cyber security benchmark IoT dataset, this white paper tests the recent machine learning (ML) approaches. The primary purpose of this study is to develop a system that can forecast also secure malware, botnets, and DDOS attacks using Honeyed styles. The goal of this architecture is to accurately represent the data and create an effective cybersecurity predictive agent. Deep Convolution Reinforcement Neural Networks (DCRNN) are used to monitor networks and classify network users as attackers. This proposed method uses a two-step network understanding skills to increase its functionality. For feature engineering issues, the first step, data preprocessing, employs DSAE (Deep Sparse Auto Encoder). The Deep Convolution Reinforcement Neural Network learning approach is used in the second step for classification. The Honeyed framework is then installed, which includes the honeyed firewall and web server. The DCRNN deployment is full, and network users can now be monitored and analyzed. The impact of the published method was evaluated using data collected in a loT environment, specifically heterogeneous data sets such as 'LITNET-2020,' 'NetML-2020, and 'IoT-23,'. Considering the statistical significance of the outcomes of this approach's assessment will be tested using state-of-the-art network detection approaches.

Keywords: Cybersecurity, Honeyed framework, DoS attack, Deep Convolution Reinforcement Neural Network, Deep Sparse Auto Encoder, IoT Environment dataset.

Introduction:

Attractions are Honeyed framework systems that trick attackers and dealing with potential attacks and real-time information about the attacks [1]. This real-time information helps security professionals identify weaknesses in the organizational network and develop stronger security policies and secure networks [2]. Depending on the activity and ability of the locus of attraction, it can be called as its interaction trap. Low-engagement attractions are functionally designed attractions with limited resources that an attacker can handle. This is only an impersonator services to the attackers [3]. Highly engaging bounties are functionally designed brawlers with all the resource an attacker can handle. It supply some realistic services and operating systems to lure intruders [4]. Honeyed framework very useful for collecting some real-time practical data about attacks, but will not be replaced firewalls and other security systems.

Honeyed framework is a collection of safety machines or machines that attract intruders. The Honeyed framework system is part of all corporate networks, but is located in a separate block. These systems have some of the data in every system to guide an attacker to initiate a hacking activity. The Honeyed framework machine continuously monitors the injected activity of the attacker's intruder. The Honeyed framework system manages proper levels of hardware devices and operating system to conduct assault monitoring tasks. The device has a massive list of threats. [5].
Honeyed frameworks create duplicate isolated networks to verify for outside communications. This supports the enhancement of the core program's security. [6]. Honeyed framework facilitates the implementation of a good strategy for Network Intrusion Detection System (NIDS). Honeyed framework-based NIDS performance may be distributed centrally or in a distributed manner. Honeyed IDS approach has been developed as part of a related study to identify several assaults. During a DoS defend another communication attack that dumps functionality of the network. Studies had made DoS IDS by grouping methods based on rules, ML and support vector machine (SVM) [7].
A knowledgeable hacker can use contemporary decryption techniques to crime at a vulnerable spot. Modifications inside the net would not be detected by the Honeyed framework, which is built on ML [8]. Here, research question that inspires the suggested solution to design a two-way Honeyed framework using a Deep RL engine [9]. This RL-supported bidirectional Honeyed framework monitors both outside and inside attackers. From these points, these Honeyed framework systems are implemented as if they were physically directly connected (ULAN Virtual Local Area Network. It supplies both internal and external people with a variety of facts that seems to be genuine. Honeyed framework suggests about this trap protects a strong layer of security between your data and the attacker [10].
The main contributions are summarized below.

1. Introducing a new trap game theory model for CPS security against attacks. It also classifies the sites of attraction into high and low interaction modes for a more precise interaction process.
2. The Honeyed framework game theory model also introduces Honeyed framework allocation and human analysis costs with limited resources, generally optimizing defense rewards for defense budgets that are not really sufficient.

Literature Survey:

Anomalous properties are of great interest. It seeks to protect itself from unauthorized use of all data and malicious intrusion onto information device. Different kind of safety remedy have been suggested for past few years, so output is still limited, [11. It's also established on ML model anomalous activity. An intelligent Honeyed framework was proposed that improves the security of IoT devices according to ML. For saving device's responses, an IoT scanner that searches the Internet for IoT devices that can be accessed from internet, and scans the internet to find out malicious interactions, improves the model called IoT Learner [12].
An independent method for attack characteristics that relies greater learning from unsupervised information collected at the Honeyed framework [13]. This approach relies on clustering procedures such as evidence accumulation, 'density-based' and 'subspace clustering', to classify traffic class flow clusters. The feature of this technique is that no training phase is needed. The Author [14] proposes the classification of network communities using social attractions for collecting information on malicious profiles.
A Honeyed framework-based defense system to overcome the limitations of existing tools [15]. Communication technology provides the management and communication of the defense system attractions and components and configures the effective communication related to process on the rule of SNMP for traffic management. For solving the issues of new threats, it depends on whether the blocks in the centripetal position to deal with the suspicious flow coming with the existing defense system. If the framework is corrupted, then the attacker is blocked by the firewall [16].
Honeypot security supply and activation in all enterprise networks play an important role in all enterprise network systems. While many strategies for detecting attacks are emerging worldwide, implementation of DL-based Honeyed frameworks is needed to configure Dynamic NIDS [17]. Attack Injection can inject various attacks to slow down your network's performance. An attacker is a user who steals or loses resources. Attacks such as Wormhole, DoS Spoofing, Blackhole, and Identity fraud is considered a huge crime in the web world. During a struggle, DoS is a form of positive assault that degrades the performance of the entire network [18].
Much research work on DoS detection method and Honeyed framework security systems has contributed. It offers designs and implementation settings for detection DoS from different network characteristics. An automated Honeyed framework system proposed to provide security features [19]. The focus is on automated network management strategies in Honeyed framework and Honeyed framework environments. The system consisted of multiple attackers with sources and double LAN. The attack events and the real-time activity of this framework were monitored by some systematic procedures. This task provides a default level of automation in honeyed environment without creating ML or DL system [20].

The implementation of IDS as well as Intrusion Prevention System (IPS) for cloud Honeyed structure [21] The system focused on malicious activities and malicious users, mainly in cloud Honeyed framework environments. In addition to this research activity, we have implemented a view of Honeyed framework mechanism for diverting harmful assaults to other isolated paths [22]. However, operation lacks deep attack analysis techniques that were less aggressive against run-time attacks.

Objective:

- Implement a Honeyed framework that uses the Cyberlion Optimization Algorithm (CLOA) to detect malware, nets, and DDOS attacks.
- The combination of deep learning and honey-net enhances the security.
- Deploy an SDN infrastructure that identifies suspicious network traffic flows and provides additional security measures to block malware, Botnet, and DDOS communications.
- Design and implementation of CADS for Honinet Firewall configured as a web server.
- Network monitoring by Deep Convolution Reinforcement Neural Network for decision making, attack and user classification.

Overview of Honeyed Framework Technology:

Honeyed framework is an integral defense system used mainly for network spoofing techniques which collect information based onthe information of hackers and some technologies as part of the intrusion detection system. Several key techniques of Honeyed frameworks: data collection, data control, and data analysis [23].
Network spoofing technology plays a key role in identifying attacks on hackers, gaining the purpose of the attack and spending a too much time and also resource on protecting the actual network [24]. Honeyed framework technology now includes multiple tricks including 'IP address spoofing', simulated system attack, network traffic spoofing, and system dynamic port configuration.]

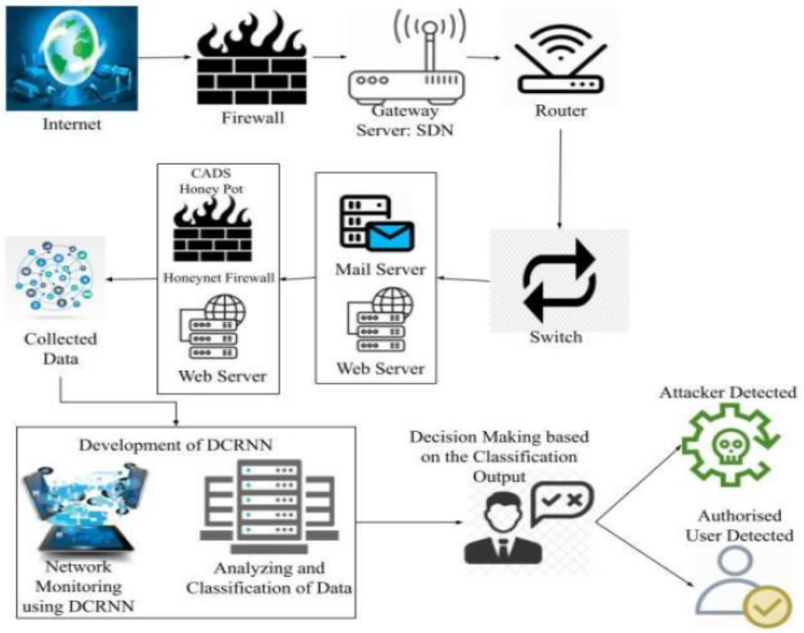


Figure: Proposed architecture

The ultimate objective of building the Honeyed framework system is to analysis the data. Honeyed framework system number analysis is mainly related to attack behavior characteristics, and the Honeyed framework collects a lot of information and does not require communication between the information, so at Honeyed framework system also has a difficult problem [25, 26, 27]. The attacker's behavior needs to build a model to analyze the data, because the data analysis module must be created to analyze the information. Data collection refers to a trap that monitors all the activities it records and designs the main unit of the trap. The key challenge is to identify as many threats as possible at the time of intrusion detection process. The lot of information gathers for analyzes the attacker's motives, strategies, and tools [28].
Data control can limit network intrusions and reduce the chances of an attacker attacking the system instead of the Honeyed framework or exploiting the Honeyed framework for harm. When an attacker enters the Honeyed framework system, every effort must be made to reduce the damage to the system, not the Honeyed framework. We can only minimize the risk [29,30]. Risk levels vary depending on the data control technology and method, but the risk cannot be completely eliminated.

The intrusion prevention system design consists of four units: central control unit, intrusion detection unit, data insertion unit and data research unit. The interface implements the control circuit. Data analysis is used to implement data analysis. Honeyed framework is used to implement the spoofing module and also the data entry module, and mechanism for finding module is to detect the data anomalies. Search the system when searching for host that is vulnerable to attack by intruders. This time, the Honeyed framework system itself, the monitoring of the misleading method used to create data transmission with outside world has been confirmed, copy some outbound traffic, and intruders are interested. Make sure that some information is a real network that is fascinated by them. Attackers use a variety of means to quickly launch attacks on the honeyed framework system for useful information and sabotage, while interacting with the implementation. The intruder modifies or deletes the task bypassing this information to the module for research methodology and the stored database as soon as possible, leaving enough time for the data recording module o record the attack information. You can prevent access to the information for. The intrusion detection module responds by looking for the best way to handle the intruder. All intruders in the particular framework system are captured by intrusion detection system, which sends early warning information to central control module, which monitors all activity of the intruder. Based on the information from the attraction system to break the security system and protect all other hosts, the data analysis module is simultaneously directed to analyze, analyze and extract the data recorded in the Honeyed framework record. Intrusion detection systems store useful rules in general databases so that intruders can attack again in the future.

Proposed Model:

The proposed Deep Convolution Reinforcement Neural Network approach for anomaly detection of network traffic data as shown in Figure. It shows a detailed architecture diagram: (1) Data-set selection, (2) Integration of data pre-processing through data distribution, and dimension reduction with deep sparse auto-encoder (3) Data output was classified as "normal / abnormal" using (4) data division, (5) stack' ensemble approach integrated with deep model and meta-learning.

Data Pre-Processing

Quite often, data collection methods generate duplicate or unnecessary attributes of network data obtained through analysis of network traffic. Removing unnecessary information, which is not important, is a step in generating a stronger representation that provides the sorter with better applicable input.
NetFlow datasets may contain mismatched properties that are included in their respective stream, core content, content, time, extra generations, and categorized features. However, the data collected from packet capture also contains a number of redundant data details, removing irrelevant information. In preprocessing, the dataset (‘IoT-23’, ‘LITNET-2020’ and ‘NetML-2020’) contains numbers of irrelevant rates. A difference of element was created to replace infinity with the maximum value.
In cases where the training data-set has an imbalance in the distribution of items, in that case ML algorithms can cause problems. The 2 established approaches, Synthetic Minority Over-sampling Technique (SMOTE) and Edited Nearest Neighbors (ENN) balance the diverse datasets. Following the application of the “SMOTE” + “ENN” to optimize some preprocessed data, we use the artificially balanced dataset to provide training for the neural network training process.
The initial layer shrinks the dimensions to 19 (‘IoT-23’), 30 (‘LITNET-2020’) and 20 (‘NetML-2020’) nodes and uses the appropriate error approximation. The next section reduce the number of features to 15 (‘IoT23’), 20 (‘LITNET-2020’) and 16 (‘NetML-2020’). In last layer, the functionality is reduced to 10 (‘IoT-23’), 15 (‘LITNET-2020’) and 12 (‘NetML-2020’), respectively. When auto-encoder training phase is successful, the network will be finalized in the input vector provider. Preliminary experiments were carried out using the test-and-error method to determine the following parameters: the weight decay $\lambda = 0.0003$, the sparsity parameter $\rho = 0.5$, and sparsity penalty term $\beta = 6$, respectively.
Dimensionality reduction:
Effects of ML sorter is directly correlated with the evaluation of the selected attributes, so removing meaningless and non-essential information is the most important step in generating stronger input for the soSrter. Principal-Component-Analysis and Auto-encoder are competent proposal for dimension reduction. The auto-encoders provide the capable to attain non-linear relations. We tested multi-layer neural networks and DAE (Deep Auto-Encoder) to reduce the number of features. Auto-encoders have the ability to formulate latent representations for input. The work included here uses three hidden-layers auto-encoder. The level uses the sigmoid role of stimulation. The input consists of n neurons based on the data-set selected after the initial pre-processing step.

Deep Reinforcement Learning:

The core of ANIDS that we have proposed is an anomaly detection engine based on reinforcement learning. It was designed to achieve its own update of the classification algorithm to operate the behavior of new network traffic, especially for new types of attacks. Unlike other IDS studies, based primarily on simulated datasets, the proposed model was designed to work in real time in a networked condition. Therefore, reliability and speed of processing are factored by the system. In our approach,

network traffic data is treated as an environmental state variable in RL, It is a processor for malware detection, the work is the same as the intrusion detection result, and the compensation is based if the outcome of the authentication is true.
Reinforcement learning (RL) is to learn by trial-and-error simulation in a dynamic environment and develop behavior. Map learning, body learning and RL are the tertiary paradigms of ML. The most important mechanism of reinforcement learning is the agent, action, the environmental state.
The process begins by performing the tasks shown in rt and state st in an environment where the agent is rewarded at time t. The state is returned to the recurrent learning agent.

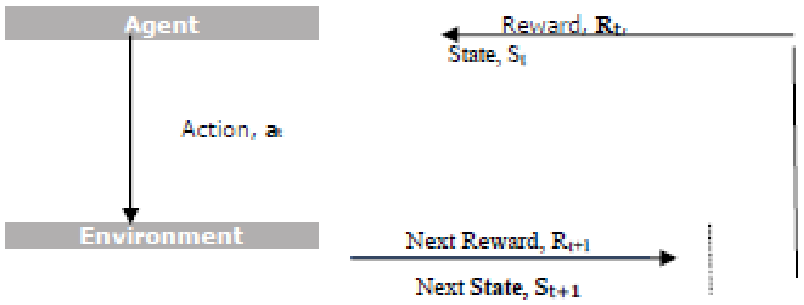


Figure: Reinforcement Learning Loop

S_t There is a trade-off between navigation and exploitation during the learning process. Agents need to leverage what they already know to change their behavioral policies to maximize rewards. The dilemma is that exploration and development cannot be done exclusively without the task being failed. Agents need to step by step to the one that looks best if they have to try different measures. In stochastic tasks, each task has several attempts to obtain reliable estimates of expected compensation. Exploration-The development dilemma has been studied by mathematicians over the decades.

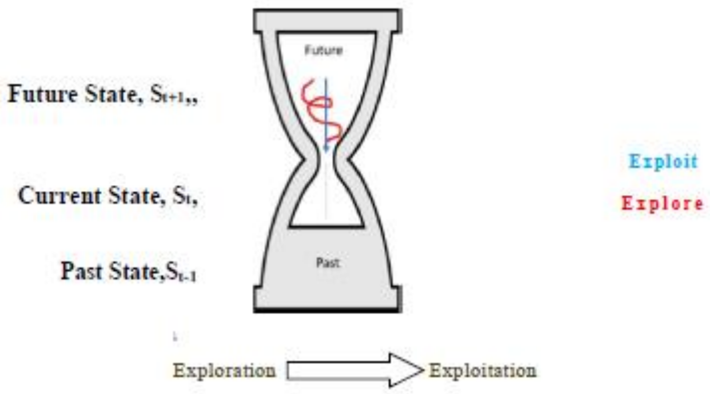


Figure: Exploration vs. Exploitation Tradeoff

Navigation percentage represent by ϵand utilization rate are represent by 1- ϵ in enforcement learning and the range of 0 towards 1. The percentage of navigation must begin with a greater possibility (often 1), and decrease as the training progresses. Whenever the training process is sufficiently mature for adequate predictive performance, agents decide on current knowledge, mostly according to abuse rate.

Deep Q-learning:

Reinforcement learning optimizes agents with the rare, time-delayed label of compensation in the environment. Markov Decision Processes (MDP) is a mathematical framework for modeling decisions using states, behaviors, and compensation. Q-learning is a strategy to find the best behavior selection policy for all MDPs. Q-Learning is a powerful policy selection algorithm but it cannot estimate unseen state values. Therefore, some Q values cannot be computed for an infinite state space where preferences can be changed. Currently, I would like to introduce Deep Q Network (DQN), which is engaged about problems of shortage of generality. There is a neural network in which current is input and the estimated Q value is output for each process.
This is an important technology that underpins a number of recent developments in detailed RL. Allowing the agent to learn in previous memory speeds up learning and can break unwanted timing connections [67]. Widely implemented in RL experiments, it was found to show excellent performance in the Actor-Critic RL algorithm, DQN, and double Q learning algorithm.

The target network for stabilizing training is reset to another network at several stages. Thus' the Q-value does not face the problem of divergence.
Figure shows the structure of this system. Reinforcement learning has two modes to achieve its own update function: Modes of learning and detection. The following is the workflow of these two different modes:

Learning Mode:

1. The RL representative processes state variables transformed from netflow statistics in its raw form to provide operations.
2. The Compensation Module calculates rewards based on actions and labels, and RL agent feedback.
3. Based on the rewards and the' status, the Reinforcement learning agent updates its policies and update intrusion detection model.

4.Continue to a first step.

Detection mode:

1. The RL agent processes state variables transformed data refers to raw internet traffic to provide operations.
2. The reward function module provides silly compensation to the RL agent and continues the process.
3. Return to step 1.

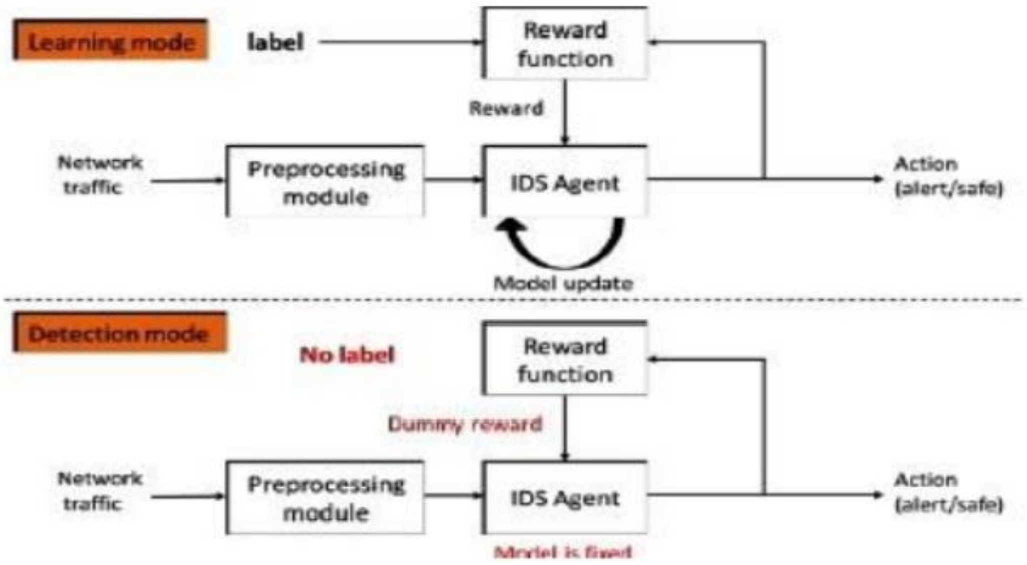


Figure: Two modes of the proposed RL

In learning mode, the DCRNN monitor rewards and assesses the validity of classification results. Reduced compensation updates the detection model as to improve on data collected intrusion detection performance. In discovery mode, the RL agent uses a fixed discovery model to handle congestion on the internet and the compensation is dummy compensation. The difference between the two mechanisms is either the compensation function uses labels to calculate the actual reward. Set the switch flag so that the system has the flexibility to switch between both modes. This feature enables the system to regularly assess and modify the search model. DCRNN agent particular tasks as a result of intrusion detection. Method can be extended for intrusion prevention work. The design of DRL is based on the DQN, that employs deep neural network to determine the estimated reward.
DCRNN purpose is for distinguish a malignant sample from attained sample, and the purpose of a malicious attack is to disguise a malignant sample as a trained sample. In this article, label Ill" indicates malignant and label "0" indicates positive.

Training Algorithm

Extract features from the original file set and name each vector.
Construct a deep neural network with optimal Adam and Loss function
Adjust the training set by batch
Set parameters
Generate parameters of malicious attack samples *o*
Adding MAS to the training set of files
Retrain the new set of files
Training algorithm
Initialize memory D playback to capacity N

```
Initialize the Q online dual system with random weight $\theta,\alpha,\beta$
Initialize the Q dueling target network with random-weight $\theta^{-}= \theta$, $\alpha^{-}= \alpha,\beta^{-}= \beta$
For episode = 1, M do
Enter the selected file ft from the sample list Preprocess ft with the Feature Extraction Module, obtain the current
kenvironment State Output Vector $S_t$
For $t$=1do.
If Pst is calculated by Eq (16), select a random action at =argmax$_a$ $Q(st, a; \theta,\alpha,\beta)$
Change ft to action and get ft+1. Get st+1 through the feature extraction module.
Enter ft+1 in DeepDetectNet and get the output label, calculate the premium rt with Eq (14)
Save transactions (st, at, it, st+1) in D
Samle random transition minibatch (sj, aj, rj, sj+1) from D
Set
$R_J = \begin{cases} rj \ \ Step\ j+1 \\ rj + yQ \end{cases}$

Where $Q = Q^{-}(S_{j+1}, argmax_a Q(s_{j+1},\ a;\theta_j,\alpha_j,\beta_j);\ \theta^{-}_j,\alpha^{-}_j,\beta^{-}_j)$
Lossj calculated in addition to e network parameters $\theta,\alpha,\beta$
Every C steps reset $\theta^{-} = \theta$, $\alpha^{-} = \alpha$, $\beta^{-} = \beta$
end if
end for
end for
```

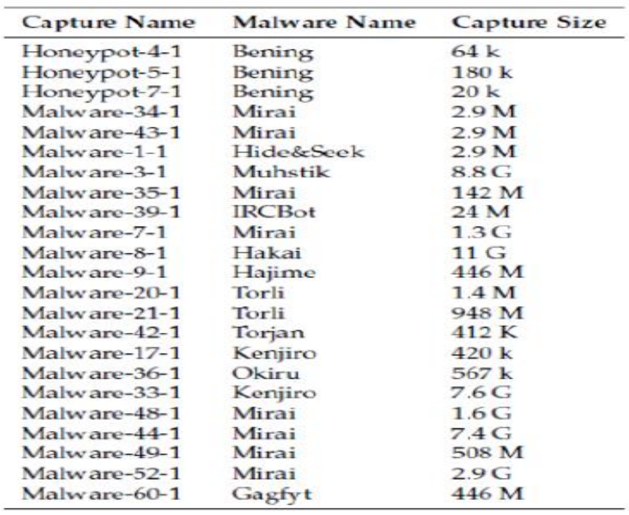
Datasets Used in Cyber-attack Detection

Data-sets using ML and optimization for classification and feature selection problems are a major factor. Because these techniques work in the learning and testing stages of learning from existing data, the datasets that various authors and scientists use to recognize how other ML can be applied. Having proper knowledge is essential. This section explains the different types of data sets used by ML and optimization algorithms in order to detect attacks.

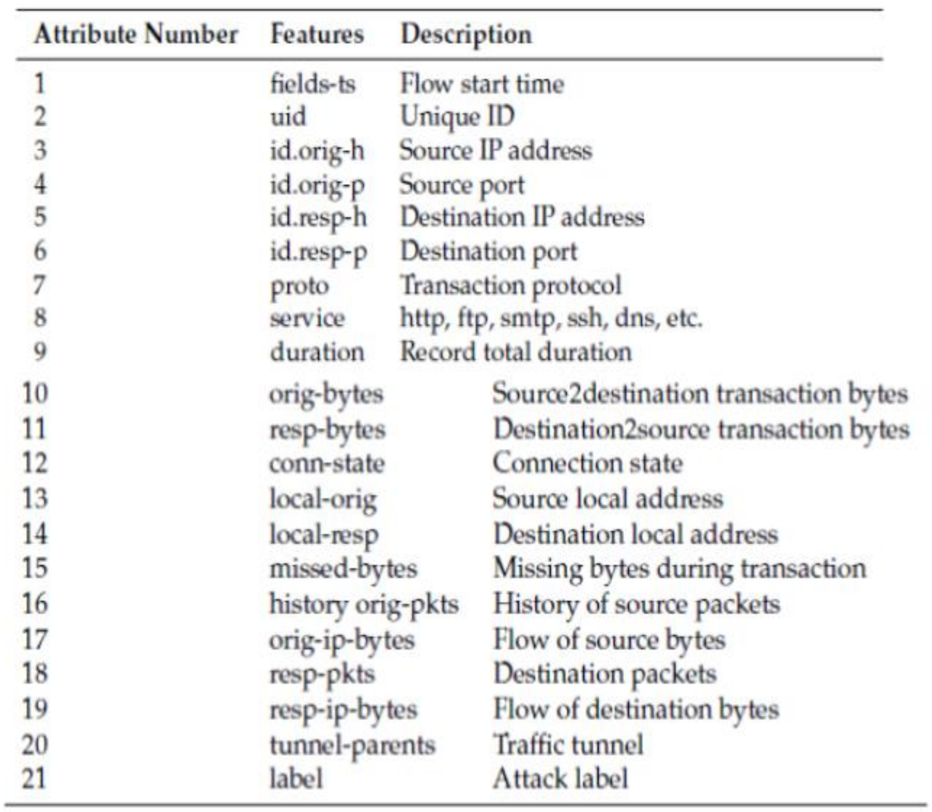
IoT-23:

'IoT-23' is a record of web traffic that integrates 20 malware subsets and 3 benign subsets. This file was first granted access at the Czech Stratosphere Institute in January 2020. The purpose of the dataset is to deliver a wide range of malware and traffic that has been classified from actual captures to developing an intrusion detection tool using ML algorithms.

But, There are 21 feature values in the collection, comprising class labels. Thence, each data instance has a total of 21 attributes that determine the characteristics of the connection. The attributes are essentially mixed, some using nominal, some numbers, and time stamp values. The 'IoT-23' dataset consists of 7 attributes ('Malware-1-1', 'Malware-3-1', 'Honeyed framework-4-1', 'Honeyed framework-5-1', 'Honeyed framework-7-1', 'Malware-34- 1', 'Malware-43-1'). Table 5 lists functional attributes together with the 'IoT-23' data set description.



Capture Name	Malware Name	Capture Size
Honeypot-4-1	Bening	64 k
Honeypot-5-1	Bening	180 k
Honeypot-7-1	Bening	20 k
Malware-34-1	Mirai	2.9 M
Malware-43-1	Mirai	2.9 M
Malware-1-1	Hide&Seek	2.9 M
Malware-3-1	Muhstik	8.8 G
Malware-35-1	Mirai	142 M
Malware-39-1	IRCBot	24 M
Malware-7-1	Mirai	1.3 G
Malware-8-1	Hakai	11 G
Malware-9-1	Hajime	446 M
Malware-20-1	Torli	1.4 M
Malware-21-1	Torli	948 M
Malware-42-1	Torjan	412 K
Malware-17-1	Kenjiro	420 k
Malware-36-1	Okiru	567 k
Malware-33-1	Kenjiro	7.6 G
Malware-48-1	Mirai	1.6 G
Malware-44-1	Mirai	7.4 G
Malware-49-1	Mirai	508 M
Malware-52-1	Mirai	2.9 G
Malware-60-1	Gagfyt	446 M



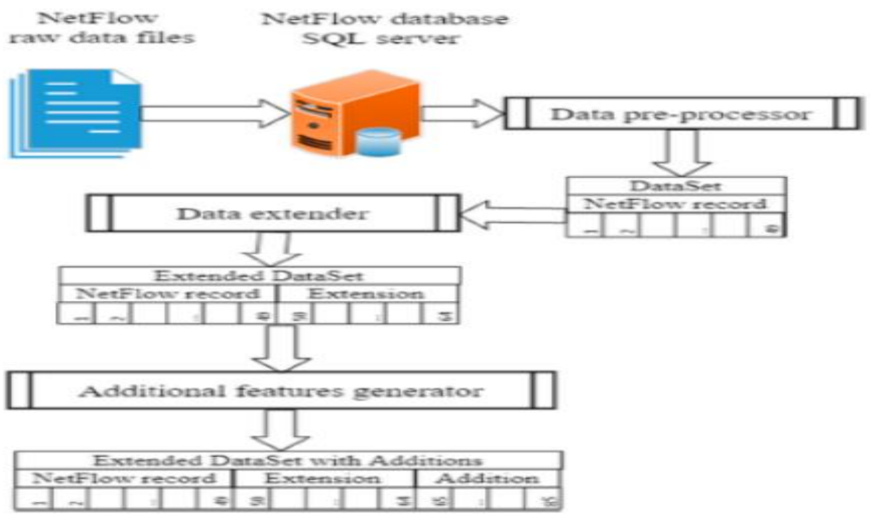
Attribute Number	Features	Description
1	fields-ts	Flow start time
2	uid	Unique ID
3	id.orig-h	Source IP address
4	id.orig-p	Source port
5	id.resp-h	Destination IP address
6	id.resp-p	Destination port
7	proto	Transaction protocol
8	service	http, ftp, smtp, ssh, dns, etc.
9	duration	Record total duration
10	orig-bytes	Source2destination transaction bytes
11	resp-bytes	Destination2source transaction bytes
12	conn-state	Connection state
13	local-orig	Source local address
14	local-resp	Destination local address
15	missed-bytes	Missing bytes during transaction
16	history orig-pkts	History of source packets
17	orig-ip-bytes	Flow of source bytes
18	resp-pkts	Destination packets
19	resp-ip-bytes	Flow of destination bytes
20	tunnel-parents	Traffic tunnel
21	label	Attack label

LITNET-2020

‘LITNET-2020’: NetFlow data-set [29] consists of caller and recorder. The caller consisted of a Cisco router and a FortiGate (FG-1500D) firewall and was leveraged to evaluate NetFlow data is sent to the collector. The recorder integrates software that explains data reception, storage, and filtering. Table shows the specific number of samples for the dataset class (including 45.492.310 streams). All cases are classified as regular data45, 330,333 streams) and malicious data (5,328,934 stream)
Based on the type of network intrusion, disruptive instances are also divided into nine classes.

Attack Type	Flows	Attacks
Smurf	3,994,426	59,479
10P-flood	3,863,655	11,628
UM-flood	606,814	59,479
TCP SYN-flood	14,608,678	3,725,838
HTTP-flood	3,963,168	22,959
LAND attack	3,569,838	52,417
Blaster worm	2,858,573	24,291
Code red worm	5,082,952	1,255,702
Spam bot's detection	1,153,020	747
Reaper worm	4,377,656	1176
Scanning/ spread	6687	6232
Packet fragmentation attack	1,244,866	477

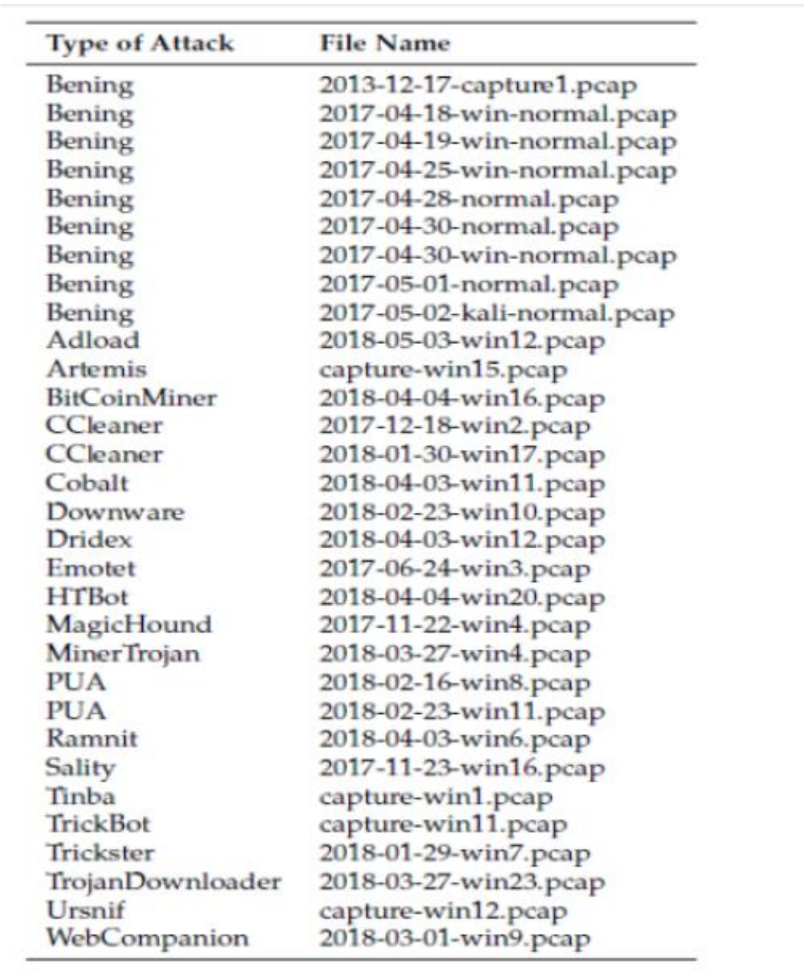
Sort the dataset by selecting the 49 features specified in the first NetFlow V9 protocol [30] of the data preprocessor. 15 different functional attributes are complemented by the data enhancer. However, 19 additional attributes are provided to recognize the type of attack. So that the final dataset has a property set of 84 different attributes.
The data set structure is summarized in Figure 3. To set the data set for the data preprocessor, select 49 specific attributes to the NetFlow v9 protocol. Data Extender extends the dataset created in the tcp tag with the additional time field to determine the attack later. A set of 15 traits completes the expanded data set. The generator generates another 19 attributes to determine attack type, built-in NetFlow capabilities, then includes two fields that differentiate records, attack type and general network traffic in the dataset. Therefore, a total of 85 attributes are available.



NetML-2020:

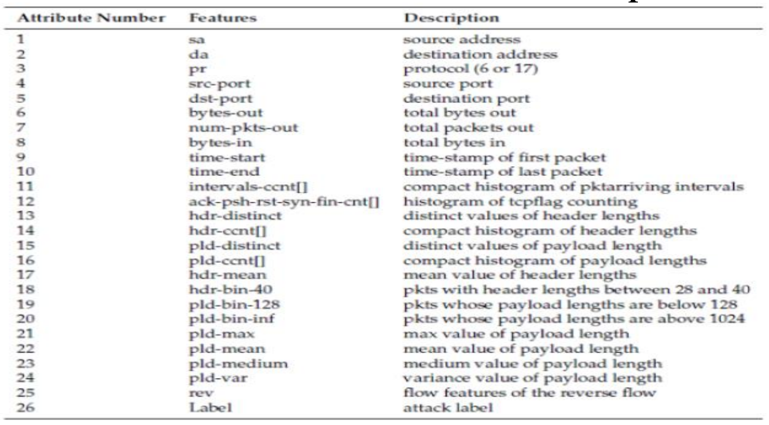
'NetML-2020' dataset gets 30 traffic data with Stratosphere IPS. It was created for the purpose of detecting anomalies. Several properties are derived in the manner of JavaScript Object Notation format providing a raw 'pcap' file as an input to the feature extraction tool and the sample of each stream is listed in the result file. There will be a unique id to identify all flow information and name information with the initial traffic packet capture file. Each data collection contains 484,0561 streams and 48 characteristics then 26 meta features are selected due to "top" granularity. Tables 8 and 9 provide functional attributes and a detailed description of the capture file selected.

Table: NetML dataset attributes



Type of Attack	File Name
Bening	2013-12-17-capture1.pcap
Bening	2017-04-18-win-normal.pcap
Bening	2017-04-19-win-normal.pcap
Bening	2017-04-25-win-normal.pcap
Bening	2017-04-28-normal.pcap
Bening	2017-04-30-normal.pcap
Bening	2017-04-30-win-normal.pcap
Bening	2017-05-01-normal.pcap
Bening	2017-05-02-kali-normal.pcap
Adload	2018-05-03-win12.pcap
Artemis	capture-win15.pcap
BitCoinMiner	2018-04-04-win16.pcap
CCleaner	2017-12-18-win2.pcap
CCleaner	2018-01-30-win17.pcap
Cobalt	2018-04-03-win11.pcap
Downware	2018-02-23-win10.pcap
Dridex	2018-04-03-win12.pcap
Emotet	2017-06-24-win3.pcap
HTBot	2018-04-04-win20.pcap
MagicHound	2017-11-22-win4.pcap
MinerTrojan	2018-03-27-win4.pcap
PUA	2018-02-16-win8.pcap
PUA	2018-02-23-win11.pcap
Ramnit	2018-04-03-win6.pcap
Sality	2017-11-23-win16.pcap
Tinba	capture-win1.pcap
TrickBot	capture-win11.pcap
Trickster	2018-01-29-win7.pcap
TrojanDownloader	2018-03-27-win23.pcap
Ursnif	capture-win12.pcap
WebCompanion	2018-03-01-win9.pcap

Table: NetML-2020 attributes with descriptions.



Attribute Number	Features	Description
1	sa	source address
2	da	destination address
3	pr	protocol (6 or 17)
4	src-port	source port
5	dst-port	destination port
6	bytes-out	total bytes out
7	num-pkts-out	total packets out
8	bytes-in	total bytes in
9	time-start	time-stamp of first packet
10	time-end	time-stamp of last packet
11	intervals-ccnt[]	compact histogram of pktarriving intervals
12	ack-psh-rst-syn-fin-cnt[]	histogram of tcpflag counting
13	hdr-distinct	distinct values of header lengths
14	hdr-ccnt[]	compact histogram of header lengths
15	pld-distinct	distinct values of payload length
16	pld-ccnt[]	compact histogram of payload lengths
17	hdr-mean	mean value of header lengths
18	hdr-bin-40	pkts with header lengths between 28 and 40
19	pld-bin-128	pkts whose payload lengths are below 128
20	pld-bin-inf	pkts whose payload lengths are above 1024
21	pld-max	max value of payload length
22	pld-mean	mean value of payload length
23	pld-medium	medium value of payload length
24	pld-var	variance value of payload length
25	rev	flow features of the reverse flow
26	Label	attack label

Setting :

This section evaluates the proposed DCRNN-based NIDS employing the methodology outlined in the past segment. Our model is implemented with a self-updating functionality to continuously detect anomalous incoming network traffic. In training mode, the model is continually updated until you stop training. Therefore, the evaluation uses the prediction result of the detection mode. We tested some experiment composed of two well-known synthetic intrusion assessment datasets, 'IoT-23','LITNET-2020' and 'NetML-2020' records. To guarantee the achievability of the proposed method, the subsequent two categories contain six basic approaches.

1) Three' different classic ML models: LSTM, mLSTM, DNN, and Stacked algorithm.

2) Result of testing of three reported approaches on the 'IoT-23','LITNET-2020' and 'NetML-2020' data-set.

Three main problems are considered for experiment:

- P1: What is a good deep learning model to extract features from URL vectors and detect anomalous requests?
- P2: How to optimize the detection model for high accuracy and low false alarms

For Pl, do some experiments with some ML and deep learning algorithms like 'LSTM', 'mLSTM', 'DNN', and Stacked algorithm.

For P2, compare it to traditional web attack detection methods.

Experimental: assessment upon IoT-23 Dataset:

This section evaluates the outcomes obtained by 'DCRNN' classifiers for individual classifiers such as 'LSTM', 'mLSTM', 'DNN', and Stacked algorithm. Each classifier has undergone a training process (eg'LSTM', 'mLSTM', 'DNN', Stacked, proposed 'DCRNN'). The mean accuracy, standard deviation, SEM, processing time and detection time were reported using cross-validation 5 as shown in Table.

Table: The overall accuracy of DCRNN on IoT-23 Dataset

Fold Validation	LSTM	mLSTM	DNN	Stacked	Proposed DCRNN
11	99.77	99.96	99.97	99.86	99.98
f2	99.76	99.91	99.92	99.96	99.97
f3	99.56	98.56	98.65	99.56	99.99
14	99.74	99.87	99.97	99.95	99.985
15	99.78	96.83	96.93	99.93	99.99
Avg. Accuracy	99.678	99.77	99.88	99.92	99.98
Standard Deviation	1.354	I .22	1.031	0.413	0.005
SEM	0.875	0.645	0.325	0.208	0.003

The proposed DCRNN achieved better accuracy of 99.98% over the conventional method Stacked is 99.92%, DNN is 99.88%, mLSTM is 99.77% and LSTM accuracy is 99.678% for classifying attacks. The four classic models of our study ('LSTM', 'mLSTM', 'DNN', and Stacked) work with similar classification patterns. All of these get higher accuracy due to their lower false positive rate. However, all showed relatively low recovery rates (about 65% to 71%), which can be caused by a large number of Negative forecasts that are not true. False voice indicates that anomalous traffic is misclassified as normal traffic.

In this circumstance, the durability of system is reduced and there is a risk of network intrusion. The proposed approach achieves a balance of approximately 99.98%% from accuracy, the Standard Deviation is 0.005 only.

Experimental assessment on the LITNET-2020 Dataset

The experiments were performed using LITNET-2020, the latest publicly available dataset published in May 2020 at the Kaunas University of Technology. To test the performance of the framework using the specified properties, the first individual classifiers of the train and the proposed DCRNN are shown on the SEM side using five cross-checks for accuracy, mean accuracy, standard deviation and Table.
Table: The overall accuracy of DCRNN on LITNET-2020 Dataset

Fold Validation	LSTM	mLSTM	DNN	Stacked	Proposed DCRNN
fl	99.86	99.86	99.91	99.94	99.99
f2	99.74	99.81	99.92	99.96	99.97
13	99.76	98.87	98.85	99.96	99.99
14	99.78	99.87	99.94	99.95	99.985
f5	99.78	96.85	96.93	99.94	99.99
Avg. Accuracy	99.78	99.88	99.92	99.94	99.99
Standard Deviation	1.37	1.22	0.416	0.016	0.000
SEM	0.22	0.12	0.016	0.008	0.000

Table shows the test results for the LITNET-2020 dataset. Likewise to IoT-23 test results, DCRNN approach gets a balance of approximately 99.99% from accuracy, recall and accuracy metrics. Compare this result with the four classical method, proposed v higher recall (about 97%), decreased precision (about 83% to 87%), and lower accuracy (about 78% to 82 6 detection patterns than the proposed DCRNN method. I'm waiting. This result indicates that the model has a multitude of false - positive results rate and tends to handle more than normal traffic. Users can be very problematic if the system administrator applies the prediction results and denies normal traffic. We conclude that the proposed DCRNN method based on the above evaluation provides promising test results in the LITNET-2020 use case.

Experimental assessment on the NetML-2020 Dataset

The proposed DCRNN for the NetML-2020 Challenges dataset is report the results in Table. The classification patterns, including DNNs and LSTMs, since DNNs are often used in multiple fields and 'LSTM's are strongly combined and it can improve their accuracy by the '5 fold cross-validation'.

Table: The overall accuracy of DCRNN on NetML-2020 Dataset

Fold Validation	LSTM	mLSTM	DNN	Stacked	Proposed DCRNN
Fl	99.57	99.67	99.76	99.86	99.98
f2	99.55	99.67	99.75	99.88	99.98
13	99.55	98.66	98.76	99.88	99.99
14	99.54	99.68	99.77	99.875	99.985
f5	99.58	96.683	96.783	99.89	99.99
Avg. Accuracy	99.558	99.677	99.78	99.887	99.984
Standard Deviation	1.554	1.236	1.192	0.143	0.003
SEM	0.705	0.625	0.536	0.065	0.002

For all the types produced good accuracy of about 99%, which means that these models have few false positive predictions. Get t e highest accuracy and accuracy rating with the DCRNN method. In the case of sensitivity, all models showed a similarity score of 99.984%. The outcome shows the efficiency of continuous method updates.

Conclusion:

NIDS monitors to detect anomalous activity and cyber-attacks to make certain security and secure communications and evidence. With this research, we propose DCRNN based methods to detect network problems. The proposed methodology can be applied in a variety of network environments, including data collection and data preprocessing steps. The form of RL can work in two modes. The learning mode is intended to provide continual exposure and update the model to maintain high detection accuracy of continuous network congestion, and the detection manner is determined to high processing speed presentation. We were also able to use all past and present data to design the IDS and detection layer to evade and detect some of these attacks. This work solves a team approach integrating the CRNN concept for an efficient distortion-based network IDS. In this paper, different methods are used for feature engineering and dimensionality reduction to achieve maximum efficiency. Simply put, the proposed framework can eliminate the problem of serving a dataset of network traffic these days and provide acceptable accuracy for detecting anomalous behavior in the desired network. The method provided by statistical significance test is used to show that the new IoT benchmark is an improvement over individual criteria classifiers, Including LSTM, mLSTM, DNN and Stacked algorithms. The accuracy of the IoT-23 dataset is 99.98%, the NetML-2020 dataset is 99.984%, and the LITNET-2020 dataset is 99.99%. We can extend implementation strategy further for future work and experiment with more sophisticated data sets.

References:

1. Hamada, A. O., Azab, M., &Mokhtar, A. (2018). Honeyed framework-like moving-target defense for secure intoperation. *In 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON),* 971-977. IEEE.
2. Kumar, G., Saha, R., Singh, M., &Rai, M. K. (2018). Optimized packet filtering Honeyed framework with snooping agents in intrusion detection system for WLAN. *International Journal of Information Security and Privacy (1.11SP), 12(1),* 53-62.
3. Veena, K., &Meena, K. (2019). Implementing file and real time based intrusion detections in secure direct method using advanced Honeyed framework. *Cluster Computing, 22(6),* 13361-13368.
4. Doshi, R., Apthorpe, N., &Feamster, N. (2018). Machine learning ddos detection for consumer intemet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW),* 29-35. IEEE.
5. Nguyen, S. N., Nguyen, V. Q., Choi, J., & Kim, K. (2018). Design and implementation of intrusion detection system using convolutional neural network for DoS detection. In *Proceedings of the 2nd international conference on machine learning and soft computing,* 34=38.
6. [6] Bingham, S. J., & Shirley, M. R. (2020). *U.S. Patent No. 10,560,434*. Washington, DC: U.S. Patent and Trademark Office.
7. Negi, P. S., Garg, A., &Lal, R. (2020). Intrusion Detection and Prevention using Honeyed framework Network for Cloud Security. In *2020 10th International Conference on Cloud Computing, Data Science &Engineering (Confluence),* 129-132. IEEE.
8. Banday, M. T., & Sheikh, S. A. (2020). Improving Security Control of Text-Based CAPTCHA Challenges using Honeyed framework and Timestamping. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC),* 704-708. IEEE.
9. Ormazabal, G. S., &Schulzrinne, H. G. (2014). *U.S. Patent No. 8,689,328*. Washington, DC: U.S. Patent and Trademark Office.
10. Shrivastava, R. K., Ramakrishna, S., &Hota, C. (2019). Game Theory based Modified Naive-bayesAlgorithm to detect DoS attacks using Honeyed framework. In *2019 IEEE 16th India Council International Conference (INDICON),* 1-4. IEEE.
11. Shi, L., Li, Y., Liu, T., Liu, J., Shan, B., & Chen, H. (2019). Dynamic distributed Honeyed framework based on blockchain. *IEEE Access,* 7,72234-72246.
12. Kaur, S., & Singh, M. (2019). Hybrid intrusion detection and signature generation using Deep Recurrent Neural Networks. *Neural Computing and Applications,* 1-19.
13. Karthick, R.R.; Hattiwale, V.P.; Ravindran, B. Adaptive network intrusion detection system using a hybrid approach. In Proceedings of the LEFF 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012), Bangalore, India, 3-7 January 2012; pp. 1-7.
14. Moustth, N.; Slay, J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf. Secur. J. Glob. Perspect. 2016,25,18-31.
15. Chora-s, M.; Pawlicki, M.; Puchalski, D.; Kozik, R. Machine Learning—The Results Are Not the only Thing that Matters! What About Security, Explain ability and Fairness? In Proceedings of the International Conference on Computational Science, Amsterdam, The Netherlands, 3-5 June 2020; pp. 615-628.
16. Liu, X.Y.; Wu, J.; Zhou, Z.H. Exploratory under sampling for class-imbalance learning. IEEE Trans. Syst. Man Cybern. Part Cybem. 2008,39,539-550.
17. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. J. Artif. Intel!. Res. 2002,16,321-357.
18. Rendon, E.; Alejo, R.; Castorena, C.; Isidro-Ortega, F.J.; Grancla-Gutierrez, E.E. Data Sampling Methods to Deal With the Big Data Multi-Class Imbalance Problem. Appl. Sci. 2020,10,1276.

19. Zhang, C.; Cheng, X.; Liu, J.; He, J.; Liu, G. Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status. J. Control. Sci. Eng. 2018,2018.
20. Xu, J.; Xiang, L.; Liu, Q.; Gilmore, H.;Wu, J.; Tang, J.; Maclabhushi, A. Stacked sparse autoencoder(SSAE) for nuclei detection on breast cancer histopathology images. IEEE Trans. Med. Imaging 2015, 35, 119-130.
21. Dutta, V.; Chora-s, M.; Pawlicki, M.; Kozik, R. Hybrid Model for Improving the Classification Effectiveness of Network Intrusion Detection. In Proceedings of the 13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020), Burgos, Spain, 18-20 September 2020.
22. Zhao, R.; Yan, R.;Wang, J.; Mao, K. Learning to monitor machine health with convolutional bi-directional LSTM networks. Sensors 2017,17,273.
23. Damasevicius, R.; Venckauskas, A.; Grigaliunas, S.; Toldinas, J.; Morkevicius, N.; Aleliunas, T.; Smuikys, P. LITNET-2020: An Annotated Real-World Network Flow Dataset for Network Intrusion Detection. Electronics 2020,9,800.
24. Claise, B.; Sadasivan, G.; Valluri, V.; Djernaes, M. Cisco Systems Netflow Services Export Version 9. Available online: (accessed on 14 August 2020).
25. Friedman, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. 1937,32,675-701.
26. Barut, O.; Luo, Y.; Zhang, T.; Li,W.; Li, P. NetML: A Challenge for Network Traffic Analytics. arXiv 2020, arXiv :2004.13006.
27. Kozik, R.; Choras, M.; Keller, J. Balanced Efficient Lifelong Learning (B-ELLA) for Cyber Attack Detection. J. UCS 2019,25,2-15.
28. Pawlicki, M.; Chora's, M.; Kozik, R. Defending network intrusion detection systems against adversarial evasion attacks. Future Gener. Comput. Syst. 2020, 110, 148-154.
29. Gandhi, U. D., Kumar, P. M., Varatharajan, R., Manogaran, G., Sundarasekar, R., &Kadu, S. (2018). 1-1IoTPOT: surveillance on IoT devices against recent threats. *Wireless personal communications, 103(2),* 1179-1194.
30. Kemppainen, S., &Kovanen, T. (2018). Honeyed framework Utilization for Network Intrusion Detection. In *Cyber Security: Power and Technology,* 249-270. Springer, Cham.