sciendo

Journal of Smart Internet of Things (JSIoT)
VOL 2023, No.02 | 146-162 | 2023
DOI: 10.2478/jsiot-2022-0010

Article

Optimal Deep Belief Network Enabled Vulnerability Detection on Smart Environment

Bzar Khidir Hussan[1], Zryan Najat Rashid[2], Subhi R. M. Zeebaree[3],*and Rizgar R. Zebari[4],

[1]Information System Engineering Department, Erbil Technical Engineering College, Erbil Polytechnic University, Erbil, Iraq

[2]Computer Network Department, Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani, Iraq

[3]Energy Engineering Department, Technical College of Engineering, Duhok Polytechnic University, Duhok, Iraq

[4]Computer Science Department, College of Science, Nawroz University, Duhok, Iraq

*Corresponding author: Subhi R. M. Zeebaree (email: subhi.rafeeq@dpu.edu.krd)

Received 30 September 2022; Accepted 21 October 2022; Issue Published 15 December 2023

Abstract: Cyborgs invoke visions of super-humans intertwined with innovative technologies able to surpass the restrictions of the human body. Conventional vulnerability detection models sustain maximum false positive rates and depend upon manual participation. Machine learning (ML) and Artificial intelligence (AI) technologies are exploited in several real-time applications, like vulnerability, malware, and software function detection, for high-quality feature representation learning. In this aspect, this study introduces a hyperparameter-optimized deep belief network-enabled vulnerability and classification (HOSDBN-VC) technique on cyborgs. The presented HOSDBN-VC model aims to detect and classify the existence of vulnerabilities. The presented HOSDBN-VC model involves a Z-score normalization approach to transform the input data into a useful format. In addition, the Hypercube Optimization Search Algorithm-based feature selection (HOS-FS) method is employed for selecting feature subsets. Moreover, a flower pollination algorithm (FPA) with a deep belief network (DBN) paradigm is applied for vulnerability and classification. The metaheuristics-based FPA is exploited to choose the hyperparameters related to the DBN paradigm appropriately. A wide-ranging experiment has been conducted to investigate the results of the HOSDBN-VD model under two databases, FFmpeg and LibPNG. The experimental outcomes implied the betterment of the HOSDBN-VD algorithm over recent approaches.

Keywords: Cyborgs; cybersecurity; vulnerability detection; deep learning; machine learning; metaheuristics.

Copyright © 2023 Journal of Smart Internet of Things (JSIoT) published by Future Science for Digital Publishing and Sciendo . This is an open access article license CC BY (https://creativecommons.org/licenses/by/4.0/)

1. Introduction

Cyborgs are human beings which include technological components for increasing their sensory or physical capabilities. In the social imagination, cyborgs frequently invoke visions of super-humans tangled together with cutting-edge technology able to surpass the human body's limitations [1-3]. The world's population is increasingly urbanized, and the issue of forming sustainable cities will grow further. There exists a lengthy history of the application of cybernetics, complex systems, systems thinking, and development and city control [4, 5]. However, several issues so far still need to be solved in the applications and deployment of such structures and methods and the potential significance of their use. The significance of cyberspace security is turning increasingly important. Yet, cyberspace is confronting a severe menace of invasion. The main reason for the majority of cyberattacks is vulnerabilities [6]. Vulnerability misused by invaders negotiate the integrity, availability, and confidentiality of information system [7, 8]. Prior recognition of vulnerability is a potential means for reducing the loss. Despite scholars' and experts' efforts, vulnerabilities remain a great barrier and would last in the long run. This could be justifiable of the truth that a rising amount of vulnerabilities have been published annually [9, 10].

Vulnerability recognition is a technique to find software vulnerabilities. Classical vulnerability recognition contains dynamic and static approaches. The growth of machine learning (ML) technology offers the newest methodologies to ease the blockages of classical techniques. Primary research guidelines include intellectual vulnerability recognition methodologies based on source code [11]. It could be classified into three categories they are using vulnerable pattern learning, software engineering metrics, and anomaly detection (AD). At the initial levels, software engineering metrics, namely code commits, software complexity, and developer activities, have been examined for training an ML method [12-14]. The motivation for this method is that it is highly vulnerable if the software is more complicated. Software engineering metrics consist of merits just like rapid and easily gained datasets, then efficacy in recall and precision has to be enhanced. Vulnerable pattern and anomaly recognition learning endeavours to improvise the recognition effects by using the semantic and syntactic information in the code. AD studies the legal programming outline from mature software code [15-18]. The resemblance or connection between the learned rules and the candidates was utilized to notice vulnerabilities. AD consists of the merit of finding unknown vulnerabilities. But the false negative and false positive was higher [19]. The paradigm learning technique studies vulnerable paradigms from clean and vulnerable samples [20]. Compared to the AD, this technique has a superior outcome on precision but is extremely reliable in the dataset quality.

This study introduces a hyperparameter-optimized deep belief network-enabled vulnerability and classification (HOSDBN-VC) technique on cyborgs. The presented HOSDBN-VC model involves a Z-score normalization approach to convert the input data into the appropriate form. Then, Hypercube Optimization Search Algorithm-based feature selection (HOS-FS) method is employed for selecting the feature subset. Besides, the flower pollination algorithm (FPA) with a deep belief network (DBN) paradigm is applied for vulnerability and classification. The metaheuristics-based FPA is exploited to choose the hyperparameters related to the DBN paradigm appropriately. A wide-ranging experiment has been conducted to investigate the results of the HOSDBN-VD model.

The rest of the paper is organized as follows. Section 2 provides the related works, and section 3 introduces the proposed model. Later, section 4 discusses the experimental results, and section 5 draws the concluding remarks.

2. Related Works

This section offers a comprehensive survey of recently developed vulnerability detection models. Li et al. [21] proposed a deep learning algorithm for detecting multi-class vulnerability, dubbed μVulDeePecker. The μVulDeePecker is the conception of code attention that could capture data which could assist in finding types of vulnerabilities, although the sample is smaller. For such

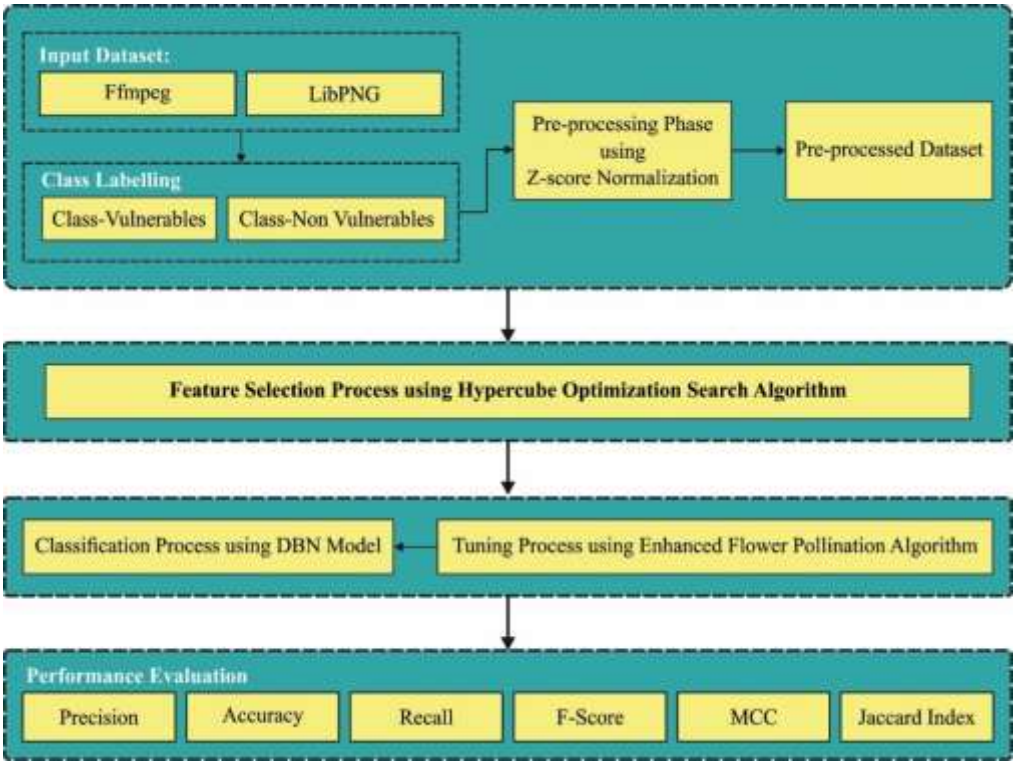
purposes, we created data from scratch and utilized them to estimate the efficiency of µVulDeePecker. Liu et al. [22] developed a DeepBalance method that integrates the novel concept of fuzzy-based class rebalancing and deep code representation learning. We designed a DNN system with Bi-LSTM for learning discriminative and invariant code representations from labelled non-vulnerable and vulnerable codes. Next, we employed a fuzzy oversampling model for rebalancing the trained dataset by creating synthetic samples for a collection of vulnerable codes.

Ban et al. [23] developed a performance assessment analysis on an ML-based solution for detecting software vulnerabilities which conducted three experimental types of research: ML-based technique for finding software vulnerabilities according to the scenario of particular kinds of vulnerabilities and different kinds of vulnerability for each data set. LI et al. [24] gathered two data sets from the programs, including 126 kinds of vulnerability, where we conducted the comparison analysis to mathematically measure the effect of diverse aspects on the efficacy of detecting vulnerabilities. Li et al. [25] developed an intelligent and automated vulnerability recognition method in source code. Firstly, the instance in the source code format is converted into an intermediate representation to exclude the irrelevant item and decrease the dependency length. Next, the vector is fed to three concatenated CNN to obtain higher-level features of vulnerability. Finally, classifiers are trained by the learned feature.

The researchers in [26] explored the graph neural network and professional expertise for detecting smart contract vulnerabilities. We cast the source code's abundant data and control flow semantics into the contracted graph. The study presents a node elimination phase for normalizing the graph. Later, a temporal message propagation method was developed to extract the graph features from the normalized graph and merge those graph features with the expertized pattern for generating the last detection scheme. Hyperparameters such as epoch count, batch size, and learning rate selection are essential to attain effectual outcomes. Since the trial and error method for hyperparameter tuning is tedious and erroneous, metaheuristic algorithms can be applied. Therefore, in this work, FPA is used for the parameter selection of the DBN model.

3. The Proposed Model

In this study, a novel HOSDBN-VC model has been introduced to detect and classify vulnerabilities in cyborgs. The projected HOSDBN-VC model employed a Z-score normalization approach to convert the input dataset into the appropriate form. Then, a novel HOS-FS method is employed for selecting a feature subset. Next, the FPA-DBN paradigm is exploited for vulnerability and classification. Fig. 1 demonstrates the workflow of the HOSDBN-VC model.



Figure

3.1 Data Pre-processing

Primarily, the proposed HOSDBN-VC methodology employed a Z-score normalization approach to convert the input data into a valuable format. The Z-score normalizing denotes the setting of each dimension data X for containing zero-mean and unit variance.

$$X_{norm} = \frac{X-\mu}{\sigma}, \tag{1}$$

Where σ and μ are the standard deviation (SD) and mean of X correspondingly. Image data has been mapped to ranges of 0 to 255 by Min-Max scaling. The conversion formula is as follows:

$$x^i = \frac{x^i - x^i_{min}}{x^i_{max} - x^i_{min}}, \tag{2}$$

In which $x^i \in X_{norm}, i \in [1,2,,n]$.

3.2 Design of HOS-FS Technique

In this phase, the pre-processed information is passed into the HOS-FS technique for choosing a feature subset. The presented approach is stimulated by an arbitrarily distributed set of points inside an m-dimension hypercube (HC) [27]. HOS exhibits faster population convergence by lessening the region of HC at all the iterations. The suggested mechanism comprises the initialization, displacement shrink, and search area processes. Such processes are briefly discussed below.

The HOS method initiates by the initialization method, where an arbitrarily produced point within an HC forms the candidate solution matrixes. Some initial conditions in the initialization stage have to be calculated, involving (1) size (r_{dim}), (2) lower-upper limitations (lb, ub), (3) dimension of the HC (m), and (4) central value (x_c).

$$m = length(x_c),$$
$$x_c = \frac{(ub + lb)}{2},$$
$$d = lb - ub, r_{dim} = \frac{d}{2},$$
$$lb = min(Xbounds),$$
$$ub = max(Xbounds). \quad (3)$$

Initially, the initial HC is formed by allocating random numbers to r_{dim} and x_c. Then, the uniformly distributed N point $x_i = (x_{i1}, x_{i2}, \dots, x_{\Im})$ are produced arbitrarily inside the HC. Also, this point is embodied in X matrix form with the dimension of $(N \times m)$. Then, the lower and upper limits of the initial HC are evaluated by the X matrix. r_{dim} and x_c of the second HC are defined. Also, the matrix X is exploited for calculation, wherein the finest value of fitness function (FF) F_{best} and equivalent x_{best} points are defined within the population at $i\text{-}th$ iterations. The x_{best} point is amended by using local search:

$$x_{best}^{new} = x_{best} + \rho \Delta F, \quad (4)$$

In Eq. (4), the FF can be represented as F, and $0 \leq \rho \leq 1$.

The displacement-shrink process focuses on determining the centre of the second hypercube (new hypercube) $x_{c_{new}}$ and calculating the FF. The center of the second hypercube can be accomplished by the average amount of the preceding hypercube centre and the existing finest point (x_{best}):

$$x_{c_{new}} = \frac{x_{best}^{new} + x_c}{2}. \quad (5)$$

Every iteration creates a novel data point and evaluates the FF. The hypercube size was adapted according to the calculation outcome. Hence, the size of HC can be reduced, and the searching space is decreased, which is named "shrinking." As soon as the hypercube size reduces, the density of the searching point rises. The movement of the finest value can be directed by shrinkage. The shrinkage is strong for small movements. This guarantees fast convergence, preventing the process from getting trapped in a local minimum. This process will cycle through a series of points beginning from the existing location to evaluate the maximal distance. Firstly, the F_{best} value is compared to the $F_{mean} = F((x_{best} + x_{last-center})/2)$. When the value of F_{mean} is lesser than F_{best} at the iteration, x displacement (or x movement) is normalized twice and computed at every iteration as follows:

Normalize x (the prior x for minimal value)

$$x_n = \frac{x - x_c}{d}, \quad (6)$$

Normalize x_{min} (existing x for minimal value)

$$x_n = \frac{x_{min} - x_c}{d}, \quad (7)$$

Normalized distance

$$d_n = \sum \quad (8)$$

Re-normalize distance

$$d_{nn} = \frac{d_n}{\sqrt{m}}. \quad (9)$$

The search area phase creates an HC by initializing a new value to r_{dim} and x_c based on the d_{nn} value. The S convergence factor is evaluated when the 0≤dnn≤1 criterion is met.

Consequently, r_{dim} and x_c values are upgraded as:

$$x_c = x_{best},$$
$$r_{dim} = r_{dim} * S, \quad (10)$$
$$S = 1 - 0.2e^{-3d_{nn}}.$$

The size of HC can be decreased by multiplying r_{dim} with $theS$ factor. When $the0 \leq d_{nn} \leq 1$ criterion is satisfied, the size of HC is unchanged. HOS ensures the faster arrival of candidate solution to global minimal by decreasing the region of the hypercube afterwards. The whole process is iterated until the end condition is satisfied.

$$Fitness = \alpha\gamma_R(D) + \beta\frac{|R|}{|C|} \quad (11)$$

Where $\gamma_R(D)$ denotes the classifier error rate. $|R|$denotes the cardinality of chosen subset, and $C \vee$ implies the overall number of features.

3.3 Vulnerability Detection and Classification

As soon as the feature was chosen, they passed into the FPA-DBN paradigm for vulnerability and classification. The DBN is a neural network, i.e., stacked by using a supervised BP network and multiple RBM networks [28]. The RBM network comprises hidden and visible layers. Given that v_i and h_j denote the initial $i\text{-}th$ neurons in the visible layer and $j\text{-}th$ neurons in the hidden state, correspondingly. For a collection of(h), the energy function of RBM is determined. Fig. 2 illustrates the structure of the DBN technique.

$$E(v_t h|\theta) = -\sum_{i=1}^{I} a_i\, v_i - \sum_{j=1}^{J} b_j\, h_j - \sum_{i=1}^{I}\sum_{j=1}^{J} v_i\, \omega_{ij} h_j \quad (12)$$

In Eq. (12), the parameters of RBM are represented as $\theta = (\omega_{ij}, \quad ai, b_j)$, the weights between the v_i and the h_j nodes invisible and hidden layers are denoted as ω_{ij}, the biased values of v_i and $h_j$are signified asa_i and b_j. The joint likelihood distribution is attained based on the energy function (v,h).

$$p(v, h|\theta) = \frac{e^{-E(v,\, h|\theta)}}{[X(\theta)]} \quad (13)$$

In Eq. (13), $Z(\theta) = \sum_v \sum_h e^{-E(v,h\vee\theta)}$ is indicated as a normalized factor. Since there was no relationship between the layers in the RBM, once the node state in the visible layers is defined, the activation state of the node in the hidden layers is mutually independent. As a result, the activation probability of the node in $thejth$ hidden layers is determined.

$$p(h_j = 1|v, \theta) = \sigma\left(b_j + \sum_{i=1}^{I} v_i\, \omega_{ji}\right) \quad (14)$$

In Eq. (14), $\sigma(x) = 1/(1 + e^{-x})$ denotes a sigmoid function. The activation probability of the node in the $i\text{-}th$ visible layer is attained at the same time.

$$p(v_i = 1|h_t\theta) = \sigma\left(a_i + \sum_{j=1}^{J} h_i\, \omega_{ji}\right) \quad (15)$$

RBM is a random NN with the activation function of a sigmoid. The variable $\theta = (\omega_i, \quad ai, b_j)$ can be attained by using constant iteration, and $\theta = (\omega_{ij,} \quad ai, b_j)$ is fitted with provided trained dataset. The variable θ is attained using the maximal logarithmic probability function on the trained subset.

$$\theta \quad = arg_\theta maxL(\theta) = arg_\theta max \sum_{t=1}^{T} l\, np\left(v^{(t)}|\theta\right) \quad (16)$$

The contrast deviation mechanism computes the logarithmic probability deviation of RBM, and the random gradient rise model is utilized for resolving the maximal logarithmic probability

function. The upgraded formula of the parameter is defined.

$$\Delta\omega_{ij} = \varepsilon\left(v_i h_j \quad _{daTa} - v_i h_j \quad _{recon}\right) \tag{17}$$

$$\Delta a_i = \varepsilon(v_i > aaTa^{-\nu} i \quad _{recon}) \tag{18}$$

$$\Delta b_j = \varepsilon\left(h_j \quad _{daTa} - h_j \quad _{recon}\right) \tag{19}$$

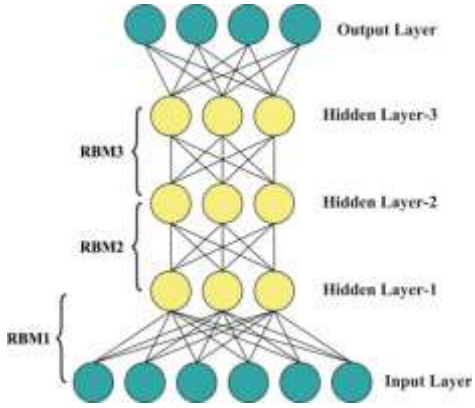


Figure 2: Structure of DBN

The above equation learning rate of pre-training is denoted as ε, the mathematical probability of the trained dataset is represented as $\cdot > data$, and the mathematical probability of the regenerated method is denoted as $\cdot \quad _{recon}$. When the output of $i\text{-}th$ nodes is 0_i, the sensitivity is δ_i, and the expectation is d_i. Next, the δ_i is defined.

$$\delta_i = o_i(1 - 0_i)(d_i - 0_i) \tag{20}$$

The formulation of sensitivity in $thel\text{-}th$ hidden layer is defined.

$$\delta_i^l = \gamma_i^l\left(1 - \gamma_i^l\right)\sum_j \omega_{ij}^l \, \delta_j^{l+1} \tag{21}$$

The updated formula of weight and bias of all the layers in the DBN in the learning rate ε is defined.

$$\omega_{ij}^l = \omega_{ij}^l + \varepsilon * y_i^l \delta_j^{l+1} \tag{22}$$

$$a_j^l = a_j^l + \varepsilon * \delta_j^{l+1} \tag{23}$$

$$b_j^l = b_j^l + \varepsilon * \delta_j^{l+1} \tag{24}$$

3.4 Hyperparameter Optimization

This study uses the FPA to choose the hyperparameter related to the DBN paradigm appropriately. FPA belongs to the family of biologically inspired MA that stimulate the characteristics of flower pollination. Pollination is a process of plant breeding, including the transmission of pollen particles receptive area of the female reproductive structure. Some agents, like wind, butterflies, bees, bats, water, and so on, carry out the transmission of pollen particles, widely called pollinators. But pollinator is classified into biotic and abiotic pollination. In biotic pollination, the pollination is performed by livestock or insects. In abiotic pollination, the pollination agent takes place by inanimate objects. Nonetheless, the pollination process may only be obtained by including pollinators. It happens once the pollen transfer from male to female portions of the same plant species is called self-pollination. The core concept of this presented algorithm is the optimum transfer of pollen that consequently promotes and increases the reproduction of flowers. The FPA replicates the flower constancy functionality and pollination

previously mentioned and mathematically modelled by four explicit rules by Yang [29]:

R_1: Cross and biotic pollination replicate the global searching method that pollinators perform in the resemblance of Lévy flight.

R_2: The local searching mimics the self and abiotic-pollination phenomena.

R_3: The flower constancy can be seen as a reproductive rate related to the similitude of two flowers.

R_3: The local and global pollination process is assumed on switching probability $p \in [O, 1]$.

The initialized population is randomly generated to define the optimal solution at the beginning of the algorithm execution. To calculate the original solution, the pollination class must be defined regarding fixed probability p, viz., arbitrary value $r \in [0,1]$. When $rand < p$, global pollination is obtained, and the following equation represents the formula for flower constancy concerning R_1 and R_2:

$$x_i^{t+1} = x_i^t + \gamma L\left(x_i^t - gbest\right) \tag{25}$$

In Eq. (1), x_i^t indicates solution i at time t, γ denotes the scaling factor, $gbest$ represents the present optimal solution, it is equated by the following expression:

$$L(s,a) \sim \frac{\lambda \Gamma(\lambda) sin\left(\frac{\pi\lambda}{2}\right)}{\pi} \frac{a}{S^{1+\lambda}},$$

$$s \gg s_0 > 0, s_0 = 0.1 \tag{26}$$

In Eq. (26), Γ denotes the standard gamma function, and the distribution is effectual for larger steps $s > 0$. Consider the second rule, viz., if $rand > p$, the local pollination procedure is performed. The flower constancy and local pollination are defined by Eq. (27):

$$\chi_i^{t+1} = \chi_i^t + \in \left(x_j^t - \chi_k^t\right) \tag{27}$$

Let x_i^t and x_k^t be pollen from a single flower of the same plant; $\in$ indicates an arbitrary integer between 0 and 1 [30]. The migration of pollinators into the favourable region is an essential part of attaining this. This area of the optimum solution is represented as the global optimal and $gbest$. Consequently, the original FPA approach can be amended by integrating the PFA system's exploratory competency to increase the pollination process and accomplish global optimal. In the following session, the process of PFA is briefly explained. The FPA system grows a FF for accomplishing better accuracy of the classifier. It resolves a positive integer for depicting the best performance of candidate outcomes. In such cases, the reduction classifier rate of errors is regarded as FF.

$$fitness(x_i) = ClassifierErrorRate(x_i)$$

$$\frac{no.ofmisclassifiedinstances}{totalno.ofinstances} * 100$$

4. Performance Validation

In this section, the simulation analysis of the HOSDBN-VC method has been tested under two databases, FFmpeg and LibPNG. The details based on the database are depicted in Tab. 1. The proposed model is simulated using Python 3.6.5 tool. The proposed model experiments on PC i5-8600k, GeForce 1050Ti 4GB, 16GB RAM, 250GB SSD, and 1TB HDD. The parameter settings are learning rate: 0.01, dropout: 0.5, batch size: 5, epoch count: 50, and activation: ReLU.

Table 1: Details on the database

Dataset Name	Total Samples	Vulnerable Samples	Not Vulnerable Samples
FFmpeg	5112	191	4921
LibPNG	542	43	499

Fig. 3 illustrates a set of confusion matrices produced by the HOSDBN-VC method on the test FFmpeg data. On the whole database, the HOSDBN-VC technique has classified 191 instances into vulnerable classes and 4902 instances into non-vulnerable classes. Also, on 70% of the training (TR) database, the HOSDBN-VC methodology has classified 146 instances into vulnerable class and 3419 instances into non-vulnerable class. Besides, on 30% of the testing (TS) database, the HOSDBN-VC system has categorized 45 instances into vulnerable classes and 1483 instances into non-vulnerable classes.

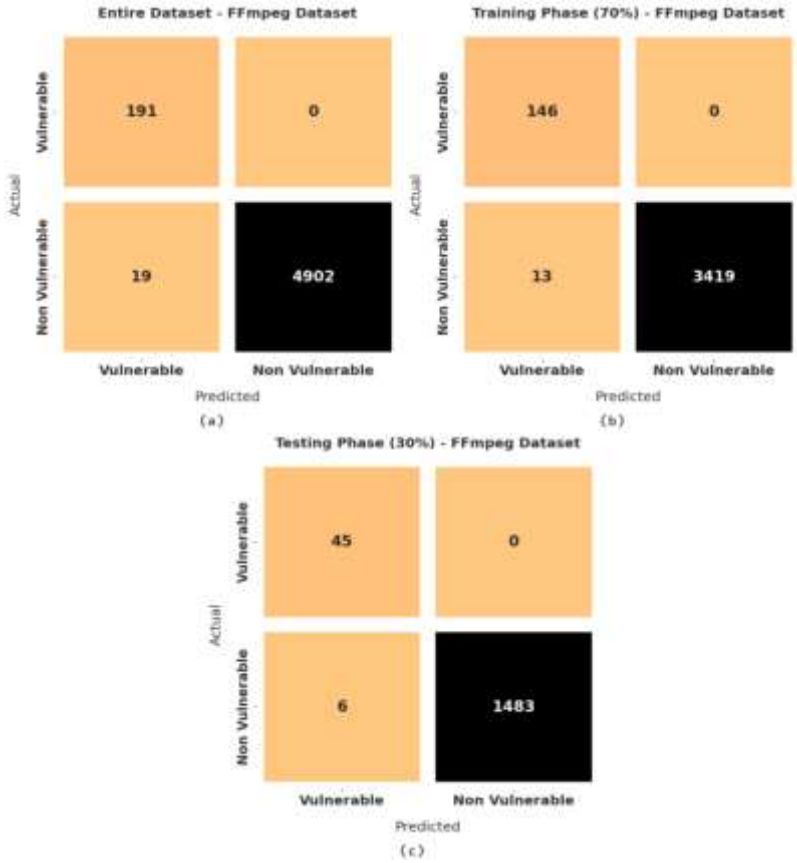


Figure 3: Confusion matrices of HOSDBN-VC technique under FFmpeg database (a) entire database, (b) 70% of TR dataset, and (c) 30% of TS database

Table 2: Result analysis of the HOSDBN-VC method with different measures on the FFmpeg database

FFmpeg Database						
Class Labels	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}	MCC	J_{index}

Entire Database						
Vulnerable	99.63	90.95	100.00	95.26	95.18	90.95
Non Vulnerable	99.63	100.00	99.61	99.81	95.18	99.61
Average	99.63	95.48	99.81	97.53	95.18	95.28
Training Phase (70%)						
Vulnerable	99.64	91.82	100.00	95.74	95.64	91.82
Non Vulnerable	99.64	100.00	99.62	99.81	95.64	99.62
Average	99.64	95.91	99.81	97.77	95.64	95.72
Testing Phase (30%)						
Vulnerable	99.61	88.24	100.00	93.75	93.74	88.24
Non Vulnerable	99.61	100.00	99.60	99.80	93.74	99.60
Average	99.61	94.12	99.80	96.77	93.74	93.92

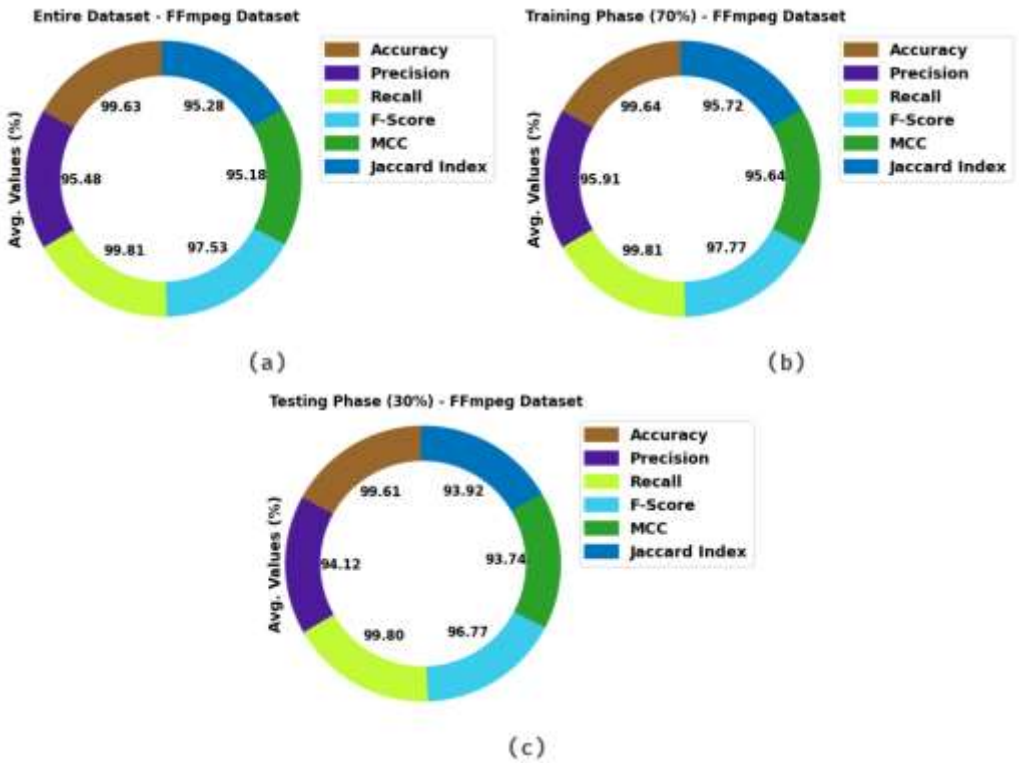


Figure 4: Average analysis of HOSDBN-VC technique under FFmpeg database (a) entire

database, (b) 70% of TR dataset, and (c) 30% of TS database

Tab. 2 and Fig. 4 depicts detailed vulnerability classification outcome of the HOSDBN-VC methodology on the FFmpeg database. The outcome indicates that the HOSDBN-VC model has gained maximum vulnerability classification performance over other techniques. For example, on the entire database, the HOSDBN-VC model has offered average $accu_y$, $prec_n$, $reca_l$, F_{score}, MCC, and J_{index} of 99.63%, 95.48%, 99.81%, 97.53%, 95.18%, and 95.28% respectively. Meanwhile, on 70% of the TR database, the HOSDBN-VC system has rendered average $accu_y$, $prec_n$, $reca_l$, F_{score}, MCC, and J_{index} of 99.64%, 95.91%, 99.81%, 97.77%, 95.64%, and 95.72% correspondingly. Eventually, on 30% of the TS database, the HOSDBN-VC method has provided average $accu_y$, $prec_n$, $reca_l$, F_{score}, MCC, and J_{index} of 99.61%, 94.12%, 99.80%, 96.77%, 93.74%, and 93.92% correspondingly.

The training accuracy (TACC) and validation accuracy (VACC) accomplished using the HOSDBN-VC system on the FFmpeg database have been depicted in Fig. 5. The result illustrates that the HOSDBN-VC method has obtained the highest values of TACC and VACC. Particularly, the VACC looked that greater than TA.

The training loss (TLS) and validation loss (VLS) accomplished using the HOSDBN-VC algorithm on the FFmpeg database are demonstrated in Fig. 6. The result illustrates that the HOSDBN-VC approach has attained lower values of TLS and VLS. Particularly, the VLS looked that lesser than TLS.

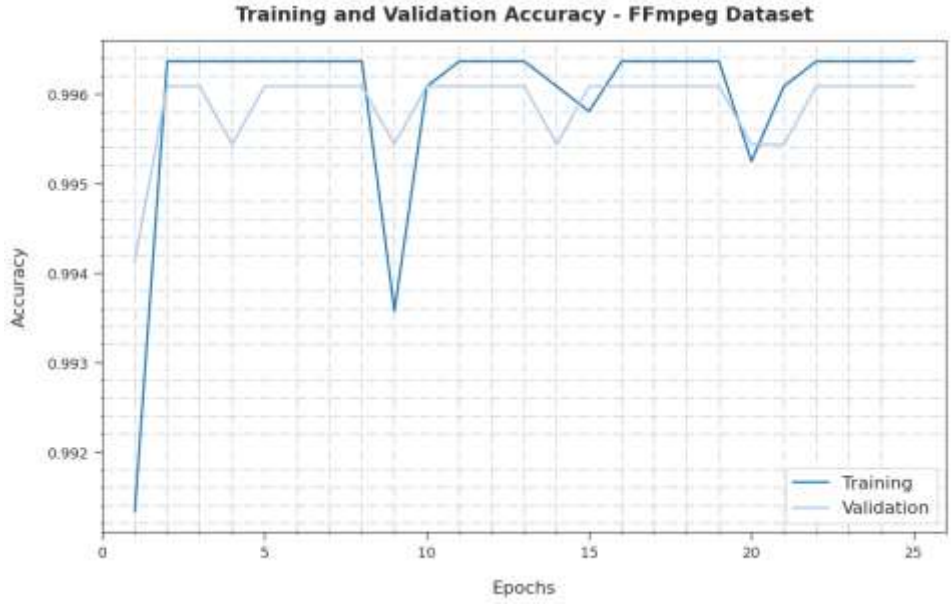


Figure 5: TACC and VACC outcome of HOSDBN-VC technique under FFmpeg database

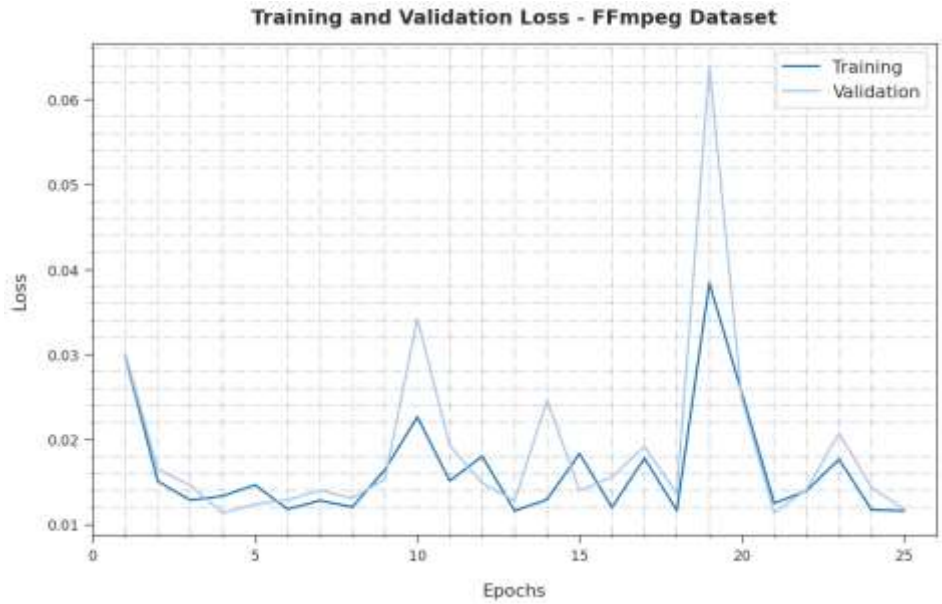


Figure 6: TLS and VLS outcome of HOSDBN-VC technique under FFmpeg database

Fig. 7 describes a set of confusion matrices produced by the HOSDBN-VC technique on the test LibPNG database. On the entire database, the HOSDBN-VC system has categorized 35 instances into vulnerable classes and 499 instances into non-vulnerable classes. Additionally, on 70% of the TR database, the HOSDBN-VC approach has categorized 25 instances into vulnerable classes and 348 instances into non-vulnerable classes. Likewise, on 30% of the TS database, the HOSDBN-VC system has categorized 10 instances into vulnerable classes and 151 instances into non-vulnerable classes.

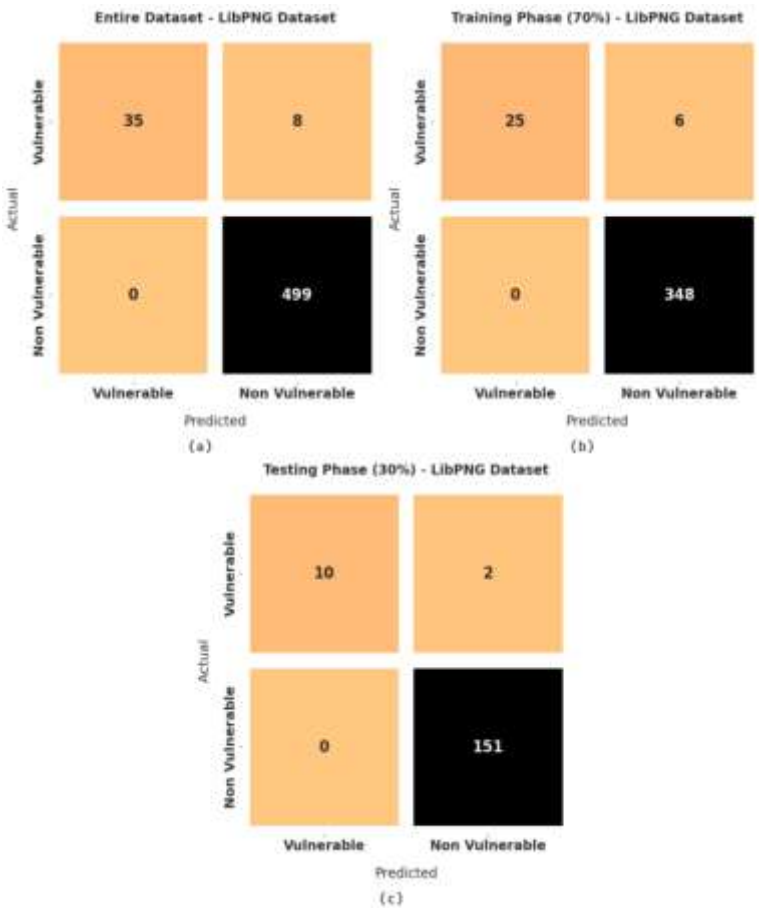


Figure 7: Confusion matrices of HOSDBN-VC technique under LibPNG database (a) entire database, (b) 70% of TR dataset, and (c) 30% of TS database

Tab. 3 and Fig. 8 offer detailed vulnerability classification outcomes of the HOSDBN-VC methodology on the LibPNG database. The outcome means that the HOSDBN-VC system has obtained maximal vulnerability classification performance over other techniques. For example, on the entire database, the HOSDBN-VC system has provided average $accu_y$, $prec_n$, $reca_l$, F_{score}, MCC, and J_{index} of 98.52%, 99.21%, 90.70%, 94.47%, 89.50%, and 89.91% correspondingly. Simultaneously, on 70% of the TR database, the HOSDBN-VC algorithm has rendered average $accu_y$, $prec_n$, $reca_l$, F_{score}, MCC, and J_{index} of 98.42%, 99.15%, 90.32%, 94.22%, 89.04%, and 89.48% correspondingly. Finally, on 30% of the TS database, the HOSDBN-VC approach has rendered average $accu_y$, $prec_n$, $reca_l$, F_{score}, MCC, and J_{index} of 98.77%, 99.35%, 91.67%, 95.13%, 90.69%, and 91.01% correspondingly.

Table 3: Result analysis of the HOSDBN-VC approach with different measures on the LibPNG database

LibPNG Database						
Class Labels	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}	MCC	J_{index}
Entire Database						
Vulnerable	98.52	100.00	81.40	89.74	89.50	81.40

Non Vulnerable	98.52	98.42	100.00	99.20	89.50	98.42
Average	98.52	99.21	90.70	94.47	89.50	89.91
Training Phase (70%)						
Vulnerable	98.42	100.00	80.65	89.29	89.04	80.65
Non Vulnerable	98.42	98.31	100.00	99.15	89.04	98.31
Average	98.42	99.15	90.32	94.22	89.04	89.48
Testing Phase (30%)						
Vulnerable	98.77	100.00	83.33	90.91	90.69	83.33
Non Vulnerable	98.77	98.69	100.00	99.34	90.69	98.69
Average	98.77	99.35	91.67	95.13	90.69	91.01

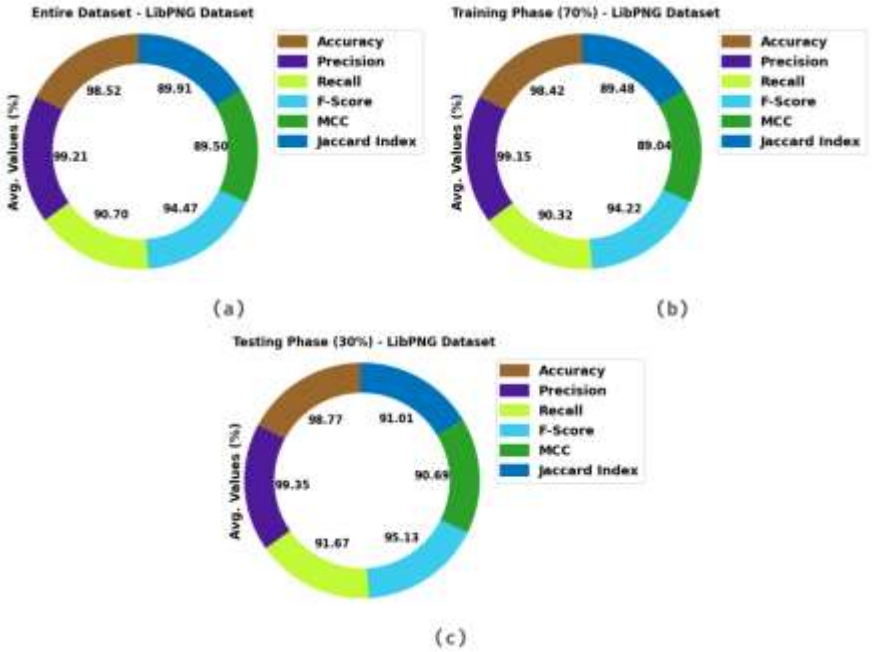


Figure 8: Average analysis of HOSDBN-VC technique under LibPNG database (a) entire database, (b) 70% of TR database, and (c) 30% of TS database

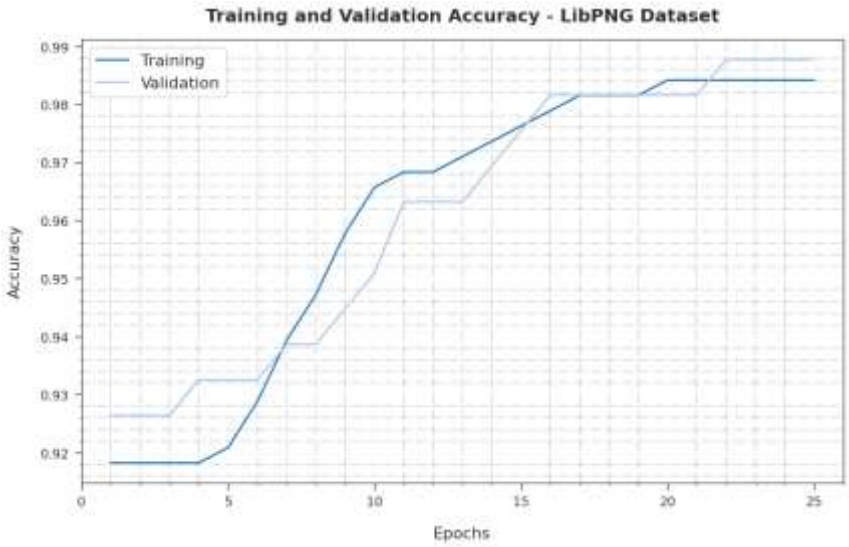


Figure 9: TACC and VACC outcome of HOSDBN-VC technique on LibPNG database

The TACC and VACC gained using the HOSDBN-VC method on the LibPNG database are illustrated in Fig. 9. The outcomes demonstrated that the HOSDBN-VC system had reached the highest values of TACC and VACC. Particularly, the VACC looked that greater than TACCC.

Finally, a comparison analysis of the HOSDBN-VC with recent approaches occurs in Tab. 4. The result ensures that the HOSDBN-VC method has enhanced results over other models. Concerning $accu_y$, the HOSDBN-VC approach has attained a higher $accu_y$ of 99.61% while the J48, KNN, LDA, NN, RF, and SVM techniques have obtained lower $accu_y$ of 89.17%, 89.36%, 88.40%, 90.54%, 93.68%, and 87.95% respectively.

Moreover, concerning $prec_n$, the HOSDBN-VC system has gained a higher $prec_n$ of 94.12%, whereas the J48, KNN, LDA, NN, RF, and SVM algorithms have acquired lower $prec_n$ of 87.61%, 87.76%, 91.37%, 89.68%, 87.89%, and 88.77% correspondingly. At last, concerning F_{score}, the HOSDBN-VC methodology has reached a higher F_{score} of 96.77%, whereas the J48, KNN, LDA, NN, RF, and SVM techniques have gained lower F_{score} of 93.48%, 90%, 93.77%, 94.12%, 93.93%, and 92.99% correspondingly. From the detailed discussion and results, it is noticeable that the HOSDBN-VC approach has attained superior outcomes over other methodologies.

Table 4: Comparative analysis of HOSDBN-VC with other approaches

Methods	$Accu_y$	$Prec_n$	$Reca_l$	F_{score}	MCC	J_{index}
HOSDBN-VC	99.61	94.12	99.80	96.77	93.74	93.92
J48 Algorithm	89.17	87.61	92.58	93.48	89.23	88.08
KNN Algorithm	89.36	87.76	93.49	90.00	89.66	89.53
LDA Algorithm	88.40	91.37	91.77	93.77	89.0	88.87

					5	
NN Algorithm	90.54	89.68	90.77	94.12	89.98	89.94
RF Algorithm	93.68	87.89	93.88	93.93	89.03	89.38
SVM Algorithm	87.95	88.77	91.16	92.99	89.21	88.60

5 Conclusion

In this study, an effective HOSDBN-VC paradigm has been introduced to detect and classify vulnerabilities in cyborgs. The projected HOSDBN-VC paradigm employed a Z-score normalization approach to convert the input data into a valuable format. Then, a novel HOS-FS method is employed for selecting a feature subset. Next, the FPA-DBN paradigm is applied for vulnerability and classification. The metaheuristics-based FPA is exploited to appropriately choose the hyperparameters related to the DBN paradigm. A wide-ranging experiment has been conducted to investigate the result of the HOSDBN-VD paradigm. The experimental outcomes implied improving the HOSDBN-VD paradigm over recent techniques with a maximum accuracy of 99.61%. Therefore, the projected method has been utilized to identify vulnerability and accomplish security effectively. In future, hybrid DL and reinforcement learning approaches can be included to improve classification performance. Besides, the proposed model can be tested on large-scale real-time data.

***Funding Statement*:** The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declare they have no conflicts of interest to report regarding the present study.

References

[1] T. Heffernan, Ed., "Cyborg futures: cross-disciplinary perspectives on artificial intelligence and robotics" Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-21836-2.

[2] Y.J. Gu, "The disembodiment of digital subject and the disappearance of women in the representations of cyborg, artificial intelligence, and posthuman," Asian Women, vol. 36, no. 4, pp. 23-44, 2020.

[3] D. Li, "Blurring human and machine boundary--the post-humanist metaphor of cyborg-body in artificial intelligence and minority report," in International Conference on Language, Art and Cultural Exchange (ICLACE), pp. 47-50, 2020.

[4] W. Shahid, Y. Li, D. Staples, G. Amin, S. Hakak and A. Ghorbani, "Are you a cyborg, bot or human?—a survey on detecting fake news spreaders," IEEE Access, vol. 10, pp. 27069-27083, 2022.

[5] Y. Khan, S. Thakur, O. Obiyemi and E. Adetiba, "Identification of Bots and Cyborgs in the# FeesMustFall Campaign," Informatics, vol. 9, no. 1, pp. 21, 2022.

[6] Z. Li, D. Zou, J. Tang, Z. Zhang, M. Sun et al. "A comparative study of deep learning-based vulnerability detection system," IEEE Access, vol. 7, pp. 103184-103197, 2019.

[7] R. Russell, L. Kim, L. Hamilton, T. Lazovich, J. Harer et al. "Automated vulnerability detection in source code using deep representation learning," in 17th IEEE international conference on machine learning and applications (ICMLA), Orlando, FL, USA, pp. 757-762, 2018.

[8] W. Wang, J. Song, G. Xu, Y. Li, H. Wang et al. "Contractward: Automated vulnerability detection models for ethereum smart contracts," IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 1133-1144, 2020.

[9] M. Goldhammer, S. Köhler, S. Zernetsch, K. Doll, B. Sick et al. "Intentions of vulnerable road users—detection and forecasting by means of machine learning," Ieee Transactions on Intelligent Transportation Systems, vol. 21, no. 7, pp. 3035-3045, 2019.

[10] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu et al. "Sysevr: A framework for using deep learning to detect software vulnerabilities," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 4, pp. 2244–2258, 2021.

[11] A. Meryem and B.E. Ouahidi, "Hybrid intrusion detection system using machine learning," Network Security, vol. 2020, no. 5, pp. 8-19, 2020.

[12] O. Achbarou, M.A. El Kiram, O. Bourkoukou and S. Elbouanani, "A new distributed intrusion detection system based on multi-agent system for cloud environment," International Journal of Communication Networks and Information Security, vol. 10, no. 3, pp. 526, 2018.

[13] D.A.A.G. Singh, R. Priyadharshini and E.J. Leavline, "Cuckoo optimization based intrusion detection system for cloud computing," International Journal of Computer Network and Information Security, vol. 9, no. 11, pp. 42, 2018.

[14] M.A. Hatef, V. Shaker, M.R. Jabbarpour, J. Jung and H. Zarrabi, "HIDCC: A hybrid intrusion detection approach in cloud computing," Concurrency and Computation: Practice and Experience, vol. 30, no. 3, pp. e4171, 2018.

[15] V. Chang, L. Golightly, P. Modesti, Q.A. Xu, L.M.T. Doan et al. "A survey on intrusion detection systems for fog and cloud computing," Future Internet, vol. 14, no. 3, pp. 89, 2022.

[16] G. Luo, Z. Chen and B.O. Mohammed, "A systematic literature review of intrusion detection systems in the cloud-based IoT environments," Concurrency and Computation: Practice and Experience, vol. 34, no. 10, pp. e6822, 2022.

[17] S. Krishnaveni, S. Sivamohan, S. Sridhar and S. Prabhakaran, "Network intrusion detection based on ensemble classification and feature selection method for cloud computing," Concurrency and Computation: Practice and Experience, vol. 34, no. 11, pp. e6838, 2022.

[18] S. Kannadhasan, R. Nagarajan and S. Thenappan, "Intrusion detection techniques based secured data sharing system for cloud computing using msvm," in 9th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, pp. 50-56, 2022.

[19] W. Elmasry, A. Akbulut and A.H. Zaim, "A design of an integrated cloud-based intrusion detection system with third party cloud service," Open Computer Science, vol. 11, no. 1, pp. 365-379, 2021.

[20] M.G. Raj and S.K. Pani, "A meta-analytic review of intelligent intrusion detection techniques in cloud computing environment," International Journal of Advanced Computer Science and Applications, vol. 12, no. 10, pp. 206-217, 2021.

[21] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin et al. "Vuldeepecker: A deep learning-based system for vulnerability detection," arXiv preprint arXiv:1801.01681, 2018, doi: /10.48550/arXiv.1801.01681

[22] S. Liu, G. Lin, Q.L. Han, S. Wen, J. Zhang et al. "DeepBalance: Deep-learning and fuzzy oversampling for vulnerability detection," IEEE Transactions on Fuzzy Systems, vol. 28, no. 7, pp. 1329-1343, 2019.