International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Zero-Day Attack Prediction a Hybrid Game Theory Approach with Neural Networks

[1]Swathy Akshaya, [2]Padmavathi G.

Submitted: 02/10/2023 **Revised**: 21/11/2023 **Accepted**: 01/12/2023

Abstract: Game theory benefits machine learning applications such as pattern recognition, photo identification, speech recognition, and intrusion detection because of its improved performance. As a result, zero-day adversarial samples are created using conventional test data and are unrecognized by the classifier, making them a more severe network threat. According to a survey most of the often used approach, there are no analytical investigations in the literature on zero-day adversarial instances that focus on attack and defense methods through trials employing a variety of settings. This study aims to apply game theory to real-life hostile circumstances, emphasizing attack and defense tactics using Modified Bi-LSTM and Game theory with ANN Auto Encoder. To do this, experiments based on gaming theory and an adaptive gaming model is used. The Nash equilibrium technique was adopted, and the standard defense mechanism was an adversarial training approach. It investigates the success rates of zero-day adversarial situations, average distortions, and original sample recognition using several adaptive game models in various settings. The research shows that adjusting the target model's parameters in real-time enhances adaptive game models' resistance to adversarial samples and the likelihood of being attacked by every node as a security metric. This method's learning mechanism is used to mimic multi-attacker information sharing.

Keywords: *Zero-Day Attack Prediction; Gaming Theory; Nash Equilibrium; Adversarial Instances; ANN, M-Bi-LSTM.*

1. Introduction

Modelling of mission-critical computer networks is very important in many fields, including defense, infrastructure, government, and the cloud [1]. In most cases, traditional security measures like intrusion detection systems and firewalls aren't enough, and managers must go elsewhere [2]. Administrators' concerns regarding hidden malware and Advanced Persistent Threat (APT) are generally driven to realize the resilience of networking over probable Zero days attacks (that is, attacks that possess exploitation of prior unrecognized/unknown vulnerabilities) [3-8]. Moreover, with the existence of standardized metrics for evaluating the relative cruelty of these unknowns. Modelling of mission-critical computer networks is very important in many fields, including defense, infrastructure, government, and the cloud. In most cases, traditional security measures like intrusion detection systems and firewalls aren't enough, and managers must go elsewhere [9-17]. These vulnerabilities are considered to be un-measurable [18]. With diverse evaluation, a promising outcome is generated in is attacking surface model, which is generally anticipated for evaluating software-based security degree along with three diverse dimensions such as channel (UDP and TCP), entry and exit points (Methods calling I/O functions), and un-trusted data factors (Configuration files and registry entries) [19]. As attack surface is significantly based on these intrinsic software factors that are specifically independent of external factors (vulnerabilities disclosure or available exploitation) may handle both unknown and known vulnerabilities and works as the finest candidate for handling threats like 0-day attacks [20-23].

1.1 Research Challenges

This study faces several research challenges in applying game theory to real-life hostile circumstances and emphasizing attack and defense tactics using Modified Bi-LSTM and Game theory with ANN Auto Encoder. These challenges include developing realistic zero-day adversarial samples that accurately mimic real-world attacks, evaluating the effectiveness of defense mechanisms against such samples through comprehensive metrics and methodologies, enabling adaptation and real-time parameter adjustment in adaptive game models, analyzing Nash equilibrium points within the game-theoretic framework, and designing a learning mechanism to effectively mimic multi-attacker information sharing. Addressing these challenges will significantly contribute to the development of robust defense strategies against zero-day adversarial attacks, enhancing the security of pattern recognition, photo identification, speech recognition, and intrusion detection systems.

[1]Research Scholar, Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women (Deemed to be University), Coimbatore, India
**Corresponding Author akshayakulandaivel@gmail.com*
[2]Professor, Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women (Deemed to be University), Coimbatore, India
Padmavathi.avinashilingam@gmail.com

1.2 Classification

Network security and adversarial machine learning are closely intertwined fields that focus on protecting computer networks and systems from malicious attacks [24-30]. Machine learning refers to the study of techniques and algorithms that aim to understand, detect, and defend against adversarial attacks on machine learning models [31-34]. These attacks exploit vulnerabilities in the models to manipulate their behavior and compromise their integrity. By understanding and mitigating these attacks, network security professionals can develop robust defense mechanisms to safeguard their systems and data [35].

1.3 Significances

The proposed methodology of applying game theory to real-life hostile circumstances, along with Modified Bi-LSTM and Game theory with ANN Auto Encoder, offers several significant contributions. Firstly, it addresses a crucial gap in the literature by investigating zero-day adversarial instances and focusing on attack and defense methods through trials encompassing diverse settings. This fills a critical research void and provides valuable insights into the effectiveness of different approaches for mitigating the threats posed by zero-day adversarial samples. Secondly, the utilization of game theory principles and adaptive gaming models allows for a proactive approach in handling zero-day adversarial attacks. By incorporating the Nash equilibrium technique and employing an adversarial training approach as the defense mechanism, the proposed methodology enhances the robustness of the target model and strengthens its resistance to adversarial samples.

1.4 Objectives

The objective of this paper is to apply game theory principles to real-life hostile circumstances, specifically focusing on zero-day adversarial instances, and investigate attack and defense methods using Modified Bi-LSTM and Game theory with ANN Auto Encoder architectures. By conducting experiments based on gaming theory and an adaptive gaming model, the study aims to assess the success rates of various adaptive game models in handling zero-day adversarial situations, average distortions, and original sample recognition across different settings. The research emphasizes the importance of adjusting the target model's parameters in real-time to enhance its resistance to adversarial samples and introduces a learning mechanism to mimic multi-attacker information sharing. By achieving these objectives, the study seeks to advance the understanding of game theory in the context of zero-day adversarial attacks and contribute to the development of robust defense strategies for machine learning applications such as pattern recognition, photo identification, speech recognition, and intrusion detection.

1.5 Significant Contributions

The paper presents significant contributions in the field of machine learning and network security. By applying game theory principles to real-life hostile circumstances, it explores the benefits of game theory in pattern recognition, photo identification, speech recognition, and intrusion detection. The study addresses the severity of zero-day adversarial samples, which pose a significant threat by being undetected by conventional classifiers. It fills a gap in the literature by providing analytical investigations on zero-day adversarial instances and examining attack and defense methods through trials in diverse settings. The research emphasizes the use of Modified Bi-LSTM and Game theory with ANN Auto Encoder architectures, conducting experiments based on gaming theory and an adaptive gaming model. The adoption of the Nash equilibrium technique and the adversarial training approach as the standard defense mechanism enhances the understanding of defense strategies. The findings demonstrate that real-time parameter adjustment improves the resistance of adaptive game models to adversarial samples, while the learning mechanism mimics multi-attacker information sharing. These contributions pave the way for the development of robust defense mechanisms and enhance the security of machine learning applications against zero-day adversarial attacks.

Obviously, along with software security, the attack surface concept has also been identified in numerous applications of other emerging domains, e.g., Moving Target Defense (MTD), cloud security, automotive security, and mobile device security. As illustrated in Fig. 1, most programmers operating at higher abstraction levels (such as Network level) can only make qualitative and intuitive sense of attack surface measurements. The initial attack surface measurements only covered a single software layer and were specified statistically and formally. Unavoidable adoption of this vague notion of loss poses unique conceptual power in quantitatively and formally reasoning with a relative probability of diverse systems enclosing vulnerabilities. Here, the issues mentioned above are resolved by elevating the original surface concept of attack toward the cloud's networking level, formally depicted with security measures such as network attack surface. This is specifically for computation of network resilience in sharing files among secure cloud storage against 0-day. The expansion of the 0-day attack surface to the network level faces two key challenges. The first mismatching aggregate score across various network resources (VM and cloudlets) is what determines the attack surface model. Subsequently, the challenging factor for eradicating cost computation of attack surface is to carry out calculation. Here, devise modeling is based on gaming theory and the Nash equilibrium approach to resolving these aforementioned confronts. The effectiveness of the anticipated model is confirmed with iterative experimentation in a cloud environment. With this experimentation, the source environment's error rate and attack surface are validated with available resources).

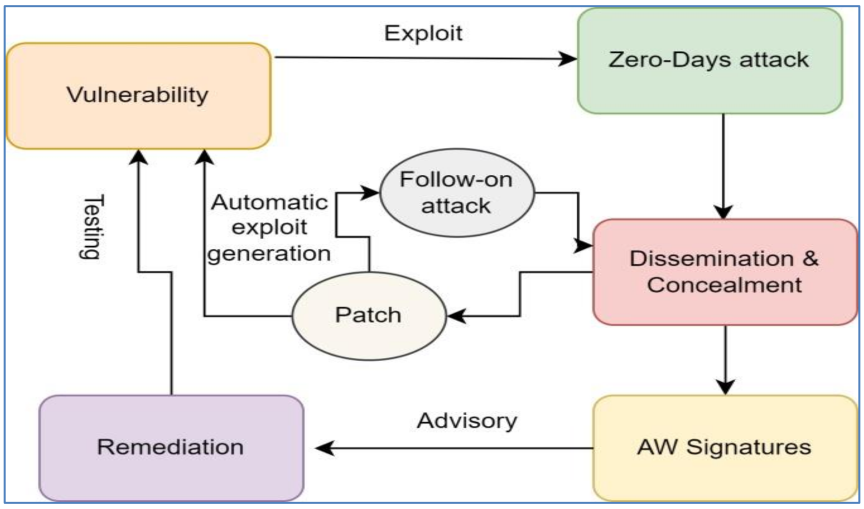

Fig 1. Zero-day Vulnerability Lifecycle

The significant involvement imposed in this investigation is twofold. Initially, this investigation has to uplift the concept of cloud attack surface by setting up a cloud environment with cloudlets and VM for data sharing and storage purposes [11]. The main objective of the work is to resolve a key restriction towards the previous investigations; that is, diverse available resources are considered to be probably equal to impose unknown vulnerabilities. In order to confirm that the expected method can provide somewhat accurate or exact results while drastically reducing the computing cost of the attack surface [12], simulation results must verify the Nash equilibrium game model's underlying assumptions. This opens the way for claims with real-world applications in networking.

The following sections of the paper are pre-arranged: Section II depicts formal models with background knowledge of Environmental worms and Background worms in the cloud. Section III designs the anticipated Gaming Theory with Nash Equilibrium. Section IV deliberates how insatiately the model works with experimental outcomes. Section V concludes the work with a future extension idea.

2. Background Study

This section discusses various investigational work carried out using zero-day attack detection. The idea behind attack surface towards software is formally anticipated as, for instance, Windows necessitates domain-specific expertise for implementation and formulation [13]. However, this idea is generalized with formal modeling and turns out to be more applicable to all software. Nonetheless, it is utilized and refined in a large-scale software environment, and corresponding computation is assisted by the generation of call graphs automatically. Over the last several years, the 0-day attack surface has been more popular. It provides the groundwork for MTD and is used as a measure for computing Kernel tailing, Android's message passing system [14]. This makes it difficult for attackers to carry out their intended purpose by fundamentally altering the attack surface within a certain period of time. The second objective is to extend the applicability of this idea to other domains, such the approximation of attack surfaces in modern automobiles and the six-way attack surfaces between services, users, and cloud-based systems [15].COPES employs byte-code static analysis for analyzing zero-day attack surfaces and protecting permission-dependent software [16]. It's also important to think about the growing tendency of automated attack surface estimation. Automatic approximation of the attack surface is performed using stack traces from user crash reports. The relationship between exposures and potential targets is also investigated [17]. Zero-day attack surface accessibility is a key factor in determining vulnerability risk. Investigation regarding the relationship between vulnerability density and attack surface is provided, even though outcomes are generally dependent on two releases related to Apache HTTP Server [18]. This shows consideration of the general existence of correlation. The 0-day attack surface inspires this sort of zero-day graph attack surface-initiated in [19]. However, it concentrates on recognizing the critical attack path set of attack graphs matched to recognize the shortest path [20]. However, other metrics are potentially applied to critical attack paths. Indeed of this attack surface, most prevailing works use higher-level abstraction, which is still restricted to informal and intuitive notions. Moreover, this is the formal attack surface metric at the networking level.

Table 1. Comparison Table

Author	Year	Methodology	Advantage	Limitation
Khoury, J., & Nassar, M.	2020	Multi-Agent Reinforcement Learning	Combining ideas from game theory and Multi-Agent Reinforcement Learning (MARL), the framework addresses CPS security on two levels: strategic and battlefield. This comprehensive approach allows for a more thorough analysis of the security landscape and enables the system to adapt and respond to evolving cyber threats effectively.	The effectiveness of the framework heavily relies on the accuracy and completeness of the models used to represent the CPS network, human administrator, malware author, and other entities involved.
Stier, J. et al. (2018)	2018	Artificial neural network	Using the Shapley value allows for the measurement of the contribution of components, such as neurons, in the neural network. This provides insights into the importance and influence of individual components in the overall functioning of the network.	The computation of Shapley values requires evaluating the contributions of each component in different coalitions, which can be computationally expensive, especially for networks with a large number of components.
Xu, J. et al.	2022	long short-term memory	The integration of prediction into the game-theoretic strategy enhances the power dispatch by accounting for the fluctuation of uncertain power load demand. By utilizing a long short-term memory (LSTM) network, the strategy incorporates velocity predictions to derive the power load demand.	The effectiveness of the strategy heavily relies on the accuracy of the velocity prediction derived from the LSTM network. If the prediction error is high, it can lead to suboptimal power dispatch decisions and potentially impact the overall performance of the energy management system.
Gao, L. et al.	2019	back propagation neural network	By using BP neural networks, the detection model can efficiently identify DoS attacks with high accuracy. Through large-scale training on the KDDCUP99 dataset, the model can select multiple feature vectors that are effective in identifying DoS attacks.	While the integration of back propagation (BP) neural networks and game theory in designing DoS attack detection methods and defense mechanisms has advantages, there are also limitations to consider

2.1 Environmental and Backend Worm

The author in [21] anticipated an anomaly-based intrusion detection approach that exploits Virtual Machine Introspection (VMI) to accumulate the sum of running modules or processes of running virtual machines. The complete information is analyzed and gathered with a centralized anomaly detector over a single virtual machine [22]. Any unrecognized module or process on an increasing number of virtual machines is measured as a Backend worm [23]. The author anticipated an approach to gathering information regarding unknown harmless procedures. Even though applying a single point to accumulate information provides a centralized abstraction view over a complete cloud network, it causes congestion among network links as the cloud network may hold thousands or hundreds of VMs [24]. Moreover, the anticipated approach cannot recognize the backend worm, which affects the trusted process.

The author in [25] anticipated detection agents in a distributed network to synchronize detection over the entire cloud network. Every agent is deployed toward the physical cloud server's virtual machine monitor (VMM). With VMI, agents gather information regarding running VMs over servers related to. Agents have to sense network functionality, VM disk image, and VM memory state. Agent’s shares information to describe threats that crosscut all physical servers [26]. The anticipated approach diminishes load over network links by distributing attack detection. Moreover, observation of all running VMs in cloud data centers is a resource consumption process and raises cooling costs and operating data centers which may diminish income profit.

The author in [27] recognized symptoms set to identify the probability of malicious activities. For all symptoms, a Forensic virtual machine (FVM) is depicted for running VMs. FVMs utilize VMI to examine running VMs from the outside environment. If the symptom is recognized in the available VM, other FVMs will be directed towards inspection of VM with other symptoms. The gathered information is transferred to the centralized module to recognize and take more action.

Author in [28], anticipated monitoring of high-interaction honey pot which analyzes memory of virtual machine running over Xen with the use of open source forensics tool Volatility. LibVMI API offers accessible volatility of VM memory with interface functions. LibVMI provides interfaces in low-level functionalities of Xen hypervisor. Modified binaries while VM inspection has been hauled out with Libguestfs, open-source libraries to manipulate and examine file systems. Even though the anticipated model succeeded in attracting an extensive range of malware, including backend worms, it will not offer an approach to gathering polymorphic worms.

In [29], the author anticipated a high-interaction double honeynet to gather polymorphic worms and produce signatures for collecting worms. Moreover, the author cast-off in-box introspection tools (tools for installing inspected VMs) like Snort-inline, Sebeka, and Honeywell. These are susceptible to attack or detection by worms. Moreover, an anticipated model cannot gather polymorphic worms that validate their presence in the target host before transmitting their code as double honey net functions by generating a loop to gather all instances of polymorphic worms.

2.2 Environmental Worm

The popular cloud computing (CC) domain from October 2007 is considered an attractive domain for various investigators. It is a novel computing approach under Google and IBM partnership. Cloud uses remote servers and the internet for maintaining applications and data. It assists users in reducing hardware usage, system maintenance, and software license. With Internet usage, users are provisioned to use cloud services over cloud applications. However, users may access services rapidly and utilize a broad network with cloud computing.CC is highly prone to environmental vulnerabilities or worms, including intruder attacks, despite these benefits. Clouds may tolerate enormous security threats and the injection of environmental worms [30]. This is measured as a serious concern. Based on the discussion above of all works, the challenging factor is considering thorough cloud examination with types of dataset and volume, detection techniques, feature selection, and analysis to recognize environmental work attacks in CC effectually. Henceforth, a novel worm-recognizing technique for dynamic analysis of VM and cloudlet with gaming theory is anticipated in this examination. This considered standard dataset KDDCup for analysis network flow to detect resilience 0-day worm while file sharing and storing. The response is also considered for analysis. It is recognized that this model will produce superior accuracy rates and least false-negative rates. These factors are discussed in section IV.

2.3 Observations of the Literature Work

The literature work provides valuable insights into zero-day attack detection. It highlights the generalization of the attack surface concept through formal modeling, making it applicable to all software. The popularity of the 0-day attack surface is emphasized, as it serves as the basis for Moving Target Defense (MTD) and enables the measurement of Kernel tailing and message passing systems in Android. The research aims to extend the applicability of the attack surface concept to other domains, such as modern automobiles and the interactions between services, users, and cloud-based systems. The use of techniques like COPES and automated attack surface estimation is explored, contributing to the analysis and protection of zero-day attack surfaces. The investigation of the relationship between vulnerability density and attack surface highlights the significance of zero-day attack surface accessibility in determining vulnerability risk. Additionally, the research introduces the concept of zero-day graph attack surface, focusing on identifying critical attack paths in attack graphs. Overall, the literature work emphasizes the need for formal and intuitive approaches to understand and measure the attack surface, especially at the networking level.

2.4 Research Gap

The aforementioned investigational work on zero-day attack detection reveals several important findings. While the idea of attack surface was initially associated with specific software domains like Windows, it has been generalized through formal modeling and found to be applicable to all software. The popularity of the 0-day attack surface has grown, serving as the foundation for Moving Target Defense (MTD) and as a metric for measuring Kernel tailing and message passing systems in Android. This hampers attackers' ability to alter the attack surface within a limited time frame. Another objective is to extend the application of the attack surface concept to other domains, such as modern automobiles and the complex interactions among services, users, and cloud-based systems. Various techniques like COPES and automated attack surface estimation are employed for analyzing zero-day attack surfaces and protecting permission-dependent software. The investigation also considers the relationship between vulnerability density and attack surface, highlighting the importance of zero-day attack surface accessibility in assessing vulnerability risk. Notably, the research addresses the need for formal attack surface metrics, including the development of the zero-day graph attack surface for recognizing critical attack paths. However, most existing works still rely on informal and intuitive notions, indicating a research gap in the field. Additionally, the formal attack surface metric at the networking level requires further exploration and development.

2.5 The Ground Works

The ground work for this study involves the recognition of the benefits of game theory in enhancing the performance of machine learning applications, particularly in pattern recognition, photo identification, speech recognition, and intrusion detection. It acknowledges the existence of zero-day adversarial samples, which are created using conventional test data and pose a significant network threat by evading classifier detection. The literature survey highlights the lack of analytical investigations focusing on attack and defense methods for zero-day adversarial instances across various settings. In response, this study aims to bridge this gap by applying game theory to real-life hostile circumstances and emphasizing attack and defense tactics through the utilization of Modified Bi-LSTM and Game theory with ANN Auto Encoder. The research employs experiments based on gaming theory and an adaptive gaming model, leveraging the Nash equilibrium technique and an adversarial training approach as the standard defense mechanism. It evaluates the success rates of zero-day adversarial situations, average distortions, and original sample recognition using multiple adaptive game models under diverse settings. The findings demonstrate that real-time adjustment of the target model's parameters enhances the resistance of adaptive game models against adversarial samples and increases the likelihood of detecting attacks from every node, thus serving as a security metric. The learning mechanism employed in this approach mimics multi-attacker information sharing, providing a foundation for further investigation and advancement in the field.

3. Material and Methods

This section illustrates in detail zero-day adversarial instances that are practically analyzed, emphasizing attack and defense techniques via experiments composed of a gaming model and an adaptive gaming model. The proposed model is represented in figure 2.

3.1 Proposed Methodology

The proposed work aims to apply game theory to real-life hostile circumstances in the context of machine learning applications such as pattern recognition, photo identification, speech recognition, and intrusion detection. By utilizing Modified Bi-LSTM and Game theory with ANN Auto Encoder, the study focuses on enhancing attack and defense tactics. The experiments employ an adaptive gaming model and leverage the Nash equilibrium technique to identify optimal strategies for both attackers and defenders. The standard defense mechanism employed is adversarial training, where the target model's parameters are adjusted in real-time to improve resistance against zero-day adversarial samples. The research investigates the success rates of zero-day adversarial situations, average distortions, and original sample recognition using adaptive game models in various settings. Additionally, the learning mechanism of the proposed method mimics multi-attacker information sharing, further enhancing the security metric. Overall, this study aims to contribute to the field by providing analytical investigations and insights into zero-day adversarial instances, offering improved performance and robustness in machine learning applications through the integration of game theory.

3.2 Data collection

The datasets are collected from Kaggle.com website https://www.kaggle.com/code/mkashifn/celosia-zero-day-attack-detection-demo/notebook. the dataset contains 8.14 GB. But we select the major CSV dataset and process the proposed workflow as dataset 1. And the dataset 2 was path dataset, the path datasets are collected from zero day attack finding from CloudSim.

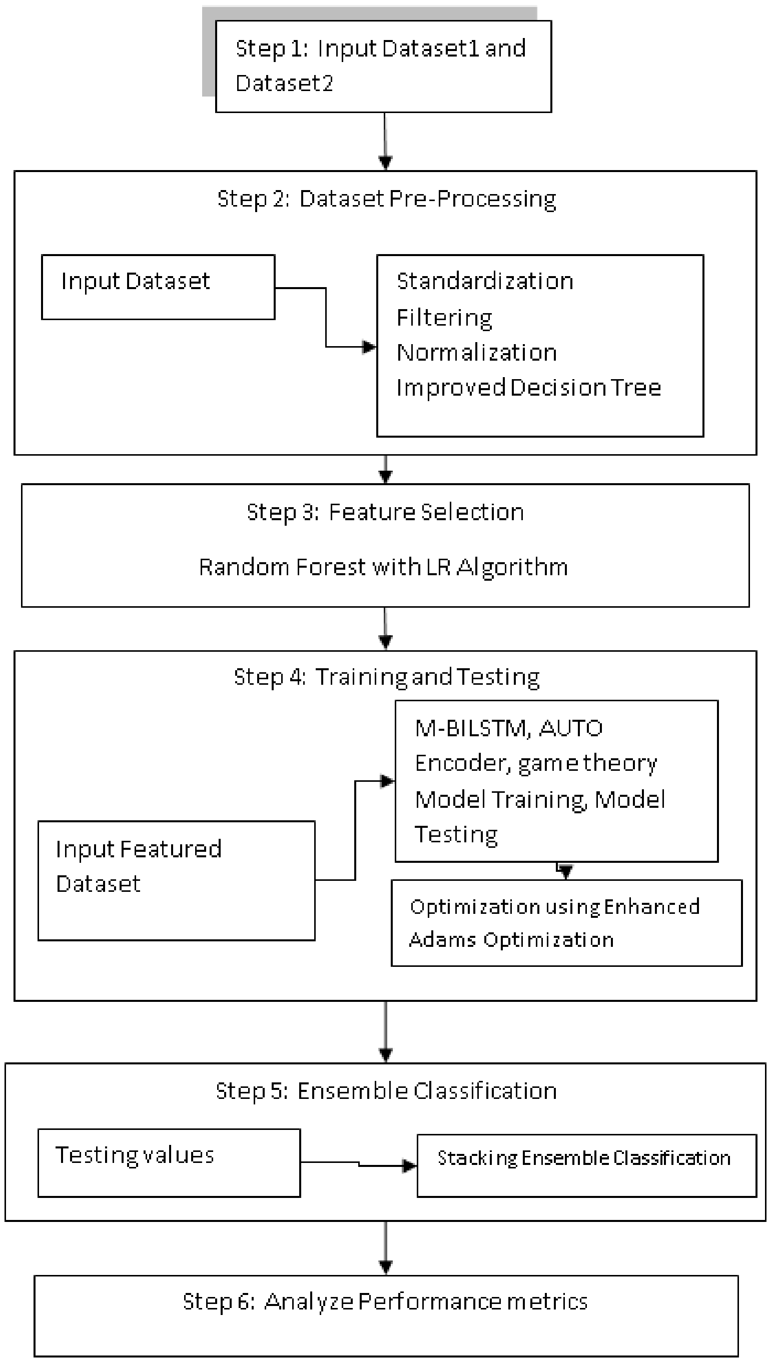
Step 1: Input Dataset1 and Dataset2
Step 2: Dataset Pre-Processing
Input Dataset
Standardization
Filtering
Normalization
Improved Decision Tree
Step 3: Feature Selection
Random Forest with LR Algorithm
Step 4: Training and Testing
Input Featured Dataset
M-BILSTM, AUTO Encoder, game theory Model Training, Model Testing
Optimization using Enhanced Adams Optimization
Step 5: Ensemble Classification
Testing values
Stacking Ensemble Classification
Step 6: Analyze Performance metrics

Fig 2. Architecture of Proposed Work

3.3 Bi-LSTM

Depending on the task at hand, auto encoders' network design may range from the simple Feed Forward network to the more complex LSTM network or Convolutional Neural Network (CNN). We have utilized CNN in this assignment. This manuscript's suggested model greatly takes use of auto encoder features. To that end, we propose using an auto encoder as a lightweight outlier detector that may be used to the problem of identifying zero-day attacks. When used in an unsupervised setting with the input as the objective, auto encoders were initially presented by Rumelhart et al. Since the input and the output are so similar, auto encoders are considered self-supervised. "A neural network that is trained to attempt to copy its input to its output" (Good fellow et al.) is the definition of an Auto encoder. Figure 1 depicts the basic layout of an auto encoder. The issue domain and intended use determine the auto encoder’s architecture and the number of hidden layers.

A function measuring the deviation between the original input x and the reconstructed input x 0 is used to characterize the reconstruction error. Mean square error (Equation 2) and mean absolute error (Equation 3) are common functions that may be used to calculate the reconstruction error.

$\mathrm{MSE} = \mathrm{N} \sum_{\mathrm{i}=0} (\mathrm{x}_0 - \mathrm{x})^2$------------------- (1)

$\mathrm{MAE} = \mathrm{N} \sum_{\mathrm{i}=0} |\mathrm{x}_0 - \mathrm{x}|$-------------------- (2)

If the encoding function f(x) is a linear function in a single-layer network, then the auto encoder is equivalent to Principal Components Analysis (PCA). Dimensionality reduction and feature learning were the first applications of auto encoders. However, several more uses have been suggested more subsequently. Word semantics, picture compression, image anomaly detection, denoising, and so on are all examples of such applications. The Auto encoder Model Suggests The suggested auto encoder is built upon an artificial neural network. Random searches are used to choose the network structure, epoch count, and learning rate for hyper-parameter optimization. Random search is believed to converge to a near-optimal set of parameters more quickly than grid search. In addition, given a constrained set of settings, it has been proven to outperform grid search. At the end, it reduces the likelihood of obtaining unrealistic parameters. Algorithm 1 explains how to train the model once the hyper-parameters have been explored.

Pseudocode- 1 Auto encoder Training

Input: Benign_data, ANN_architecture, regularisation_value, num_epochs

Output: Trained Auto encoder

```
1. training = 75% i ∈ benign_data
2. testing = benign_data ∩ training
3. Autoencoder←build_autoencoder (ANN_architecture, regularisation_value)
4. batch_size ← 1024
5. Autoencoder. Train(batch_size, num_epochs, training, testing)
6. return auto encoder
```

To begin, we divide the good examples into two groups, 75% for training and 25% for validation. After that, we apply the optimal ANN design to the initialization of the model (with regard to the number of layers and the number of hidden neurons per layer). Finally, the model is trained for n iterations. Auto encoder convergence is ensured by keeping an eye on the loss and accuracy curves. The evaluation of the model is performed using Algorithm 2.

Pseudocode 2

Input: Trained Autoencoder, attack, thresholds

Output: Detection accuracies

```
1. detection_accuracies ← {}
2. Predictions ← model. Predict(attack)
3. for th ∈ thresholds do
4.accuracy←(mse(predictions, attack)>th)/len(attack)
5.detection_accuracies.add(threshold, accuracy)
6. end for
7. return detection_accuracies
```

If the difference between the encoded instance (x 0) and the original instance (x) is more than a certain threshold, it is considered a zero-day attack. There are many cutoffs considered (.05, .1, and .15). These limits are determined by the best setting for the random search hyper-parameter. The threshold is essential in determining what MSE between x 0 and x is secure, and hence whether value signifies a zero-day attack.

3.4 Signature Module

The signature module is ultimately put to use when it comes to identifying an assault that is very context-dependent. The attack signature determines the kind of activity setting, since attacks are regarded of as a sequence of actions with preset goals. In this section, we have profiles including several attack signatures. Conditional entropy is used to create these profiles. As a tool for discerning the level of uncertainty around assaults as a function of the presence or absence of a specific event, conditional entropy is made available in the field of intrusion detection. Conditional entropy may be used to produce attack fingerprints based on observable patterns in training connections. It is dependent on the conditional likelihood of assaults given the presence of certain feature value conditions. The number of attacked files, the kind of service, and the total number of shells are all shown in Table I. Conditional probabilities characterize the degrees of co-occurrence between assaults and feature values. Probability values are higher during a conditional attack, while probability values are lesser during conditional features. Attacks reduce ambiguity, thus they are generally a good thing. Conditional computation of entropies makes use of probability values. Ultimately, conditional value entropy of specific attacks on conditional value occurrence of features is utilized in generating attack signatures. For instance, $\mathrm{Guesspassword}$ frequently occurs with telnet services. Conditional entropy encountered in telnet services ≈ 0. Moreover, this is one of the features. Therefore, features will produce extremely low conditional entropy and be used to create signatures that are provided as feature value pairs. Here, various features are created at the attack signature. An example is given below:

$[\mathrm{service}_{\mathrm{is(telnet)}}, \mathrm{flag}_{\mathrm{in(rsto,rstr)}}, \mathrm{compromised}_{\mathrm{Is(0)}}$,-- (3)

$\#\mathrm{ofshell}_{\mathrm{is(0)}}, \#\mathrm{Root}_{\mathrm{is(0)}}, \#\mathrm{ofaccessedfiles}_{\mathrm{is(0)}} \rightarrow \mathrm{Guesspassword}]$ ----------- (4)

Run time signature matches with features if, in connection record, and there should be at least one matched value of the feature. Attack alert is raised when a feature connection triggers the signature. The Anomaly detection module will process the connection when no signature is triggered.

3.5 Gaming Model

This work analyzes a defending scenario between a zero-day attack and the targeted system. The target may be an organization/company/software that includes numerous subsets. Consider network connectivity with a minimum of one cyber resource. These vulnerable nodes are under administrator protection; therefore, all subsets share a similar defense mechanism. Here, defending mechanism is performed with a combination of nodes and resources. If attackers own more than one available resource, it is involved in a defense game. This is mapped as many to many mapping. It consumes a certain period and is measured as a long-term process. If the administrator did not complete an attack by one tick, it would be performed next time. The probability of nodes attacked and corresponding metrics are computed. Here, game formulation and certain key factors are concentrated on learning strategy.

3.6 Defending Game

Resources: 0-day exploits are measured as cyber resources to attackers. Cyber resources assist attackers in completing an attack in a particular period. If one or more exploits expire, it leads to attack failure and shows that resources are failed.

3.7 Long-Term Defend

Consider an attack defense scenario with $'\mathrm{m}'$ attackers, $'\mathrm{n}'$ cyber resource, and $'\mathrm{q}'$nodes. While commencing the game, nodes are constant. However, $'\mathrm{m}'$ and $'\mathrm{n}'$ may vary from time to time. Here, the total resources are concentrated contrary to several attackers. $'\mathrm{n}'$Resources specify $'\mathrm{n}'$ attack defense games. Let $\mathrm{n(t)}$ specify the total amount of time for computation. Two factors influence computation time. Initially, $\mathrm{n}_{\mathrm{new}}(\mathrm{t})$ is resource joint newly $\mathrm{n}_{\mathrm{exp}}(\mathrm{t})$ is a number of expired resources. Thirdly, based on defense capability, some resources will be randomly eliminated during time $'\mathrm{t}'$ and depicted as $\mathrm{n}_{\mathrm{dis}}(\mathrm{t})$. Therefore; the total amount of attack defense game during time $'\mathrm{t}+1'$ is provided as in Eq.

(1):$\mathrm{n}(\mathrm{t}+1) = \mathrm{n}(\mathrm{t}) + \mathrm{n}_{\mathrm{new}}(\mathrm{t}) - \mathrm{n}_{\mathrm{dis}}(\mathrm{t}) - \mathrm{n}_{\mathrm{exp(t)}}(1)$ ----------------- (5)

Long-term defense duration: Generally, a 0-day defense game is considered a long-term game. It comprises numerous sub-sets (sub-games) over a period. The duration of gaming time is longer as resources remains to be valid. There exist three significant chances to terminate the game:

1) Resource expiration: Notation $'\mathrm{L}'$ is used to specify the lifetime of resources. Various vulnerabilities

possess diverse lifetimes. For instance, SQL injection, PHP vulnerability, and other vulnerabilities hold a higher lifetime than executable code and buffer overflow. If this vulnerability expires, the resource will also be expired.

2) Resources are found and patched due to non-existence. This is termed as passive defending ability of administrators and specified asP_a; it is depicted as discovering the probability of vulnerability before injection in resources.

3) Resources are found and patched when there is no action. This is termed an initiative defending the administrator's ability and specified as P_b, and it is depicted as discovering the probability of vulnerability before injection in resources.

Generally, P_a and P_b are provided based on administrator ability. These two values are fixed, while the administrator has precise knowledge of these fixed values; however, attackers are unknown to these values during the game process. Attackers generally assess these two fixed values, $P_a^i(t)$ and $P_b^i(t)$.after every sub-game, attackers will update their assessment by sensing the neighborhood state and exchanging information. This will be constant for all values, while attackers believe that information is free. Attackers believe that every individual has the right to access information. Therefore, attackers form a unique blacking system to share collected information effectively. From the analysis done by Baidu Security Lab (search engine), the data-sharing mechanism of the blacking system is superior to the whitening system.

3.8 Sub-Game Strategy

This is similar to a one-shot game, which is provided as the attacker's strategy to attack resources as$\{\text{anattack}, \text{not} - \text{attack}\}$, while defending set is provided as $\{\text{defending}, \text{not} - \text{defend}\}$.

3.9 Sub-Game Induction

For all available resources, $'i'$, in time $'t,'$ sub-game induction has three aspects: one-time attack $g^i(t)$, the attack cost C_a^i, long-term expenses of game $E^i(t)$. $E^i(t)$is based on four aspects like$g^i(t), P_a^i(t), P_b^i(t)$, and attackers' action in the final sub-game. While making the final decision, attackers have to consider a one-time attack and long-term expectancy for the entire lifecycle. From an administrator's viewpoint, a loss is measured in two factors, loss due to attack and defending cost. To diminish the complexity of this model, it is considered that attack loss is equal to negative attack revenge.

3.10 Rule Setting

For all sub-game with provided induction, both administrator and attacker have to make decisions by computing Nash Equilibrium. If resources remain at the end of all sub-games, attackers have to identify the corresponding neighborhood for assessment and data sharing to recomputed induction of the next commencing game. Some values in the gaming model are fixed, and they are C_a^i, C_d, P_b, L computes the following factors.

3.11 Gain Function

In some resources, the gain function specifies a one-time incoming attack, i.e., it gets access to attack the nodes at a specific time 't.' Here, $g^i(t)$ is considered a fixed value and depends on the targeted type. For instance, if an attacker needs to attack an e-commerce platform, they acquire rewards like 'Black Friday.' While in banking-based attacks, revenue over attack on weekdays is more incredible than on weekends. This shows that attack during working time provides higher revenue than off-hours attack. On the contrary, revenue is higher on late nights and weekends in the case of the video platform. Generally, consider that $g^i(t)$ is influenced by business process, visiting time, work schedules of a target. It is identified that the administrator and attacker have connectivity to the target during an attack; therefore, $g^i(t)$ values are known.

3.12 Defending Ability

When the game commences, attackers are unaware of P_a and P_b values. However, they have their assessment regarding these values as $P_a^i(t)$and $P_b^i(t)$.during the end of all sub-game, attackers have to revise the assessment value by sensing the neighborhood's state and sharing information. Rules updating is as follows:

1) Random assessment initialization during game commencement: initialization is random generally as attackers have some idea regarding the defender.

2) Computing assessment value: during end time $'t\ \ ,'$ attackers sense neighborhoods and count them by recognizing who are all attacked and who are all not attacked by them. From this observation, the values of $P_a$and P_b.

3) Merging observed value: Merging previous value with the newly observed value. By merging this, differences in reference values have to be computed. When the sensed neighborhood lives for a longer time, its reference value is considered higher. However, while commencing the game, the observed value is significant. Moreover, the observation value is unimportant as the game continued for a longer time. Due to this observation, the neighborhood of attackers is limited. Thus, observed results show more substantial randomness.

Consider that some attacker probability is being eliminated after completion of attack, P_{aD}, therefore $P_{aD} = P_a * P_D$. Alike, P_{bD} is the probability of elimination devoid of any attack. For attacker $i \in AT$ at endpoint $t \in [0, L_i]$, $P_a^i(t), P_b^i(t)$ is an assessment related to P_a and P_b. Initial assessment evaluation is $P_a^i(0)$and$P_b^i(0)$. Then, the neighborhood state is measured with $s = \{s_1^t, s_2^t, \dots, s_k^t\}$, while $f = \{f_1^t, f_2^t, \dots, f_k^t\}$ is to measure whether the neighborhood is a defender or not, where the total amount of neighborhood is depicted as k_t, that is, $s_j^t \in S_a, f_j^t \in 0,1$. The number of neighborhoods attacked during game time is specified as AD. Thus, $AD = \{j | s_j^t = 0 \wedge f_j^t = 1, j \in [0, k_t]\}$. While, neighborhood whom attackers do not attack are provided as $ND = \{j | s_j^t = 0 \wedge f_j^t = 1, j \in [0, k_t]\}$. P_D of neighbors is computed with $P_D^t(j)$. Observed value of $P_{obsAD}^t$and P_{obsBD}^t is given by $P_{obsAD}^t = |AD| / \sum s_j^t$; $P_{obsBD}^t = |ND| / (k - \sum s_j^t)$.

Now, recording the computation of neighbors is provided by Eq. (2) & Eq. (3):

$$P_{AD}^{t+1} = \frac{\sum_{j=1}^{k_t} P_a^t(j) P_D^t(j) + P_a^t(i) P_D^t(i) + P_{obsAD}^t(i)}{k_t + 2} \text{ -------- } (6)$$

$$P_{AD}^{t+1} = \frac{\sum_{j=1}^{k_t} P_D^t(j) + P_D^t(i)}{k_t + 1} \text{ -------- } (7)$$

Then, the resource life cycle has to be analyzed based on P_a and P_b, then P_{aD}^{t+1}andP_{bD}^{t+1}. Long termed games will be estimated based on a certain stage. The estimated value is nearer to the actual value; however, some uncertainty is encountered in observation. These $P_{obsBD}^t(i)$ and $P_{obsAD}^t(i)$ are removed then.

Next, neighborhood weight has to be estimated where some neighbors are no longer estimated due to reference value. $P_a^t(i)$And$P_b^t(i)$is resource time in the network, but some neighborhoods doing not exist. Here, $T = \{t_1, t_2, .., t_k\}$ specify $'k'$ existence time of neighbors.

3.13 Gaming Security with Nash

Here, the induction matrix for all sub-games is provided as in Eq. (8):

$$G_{NA}^{ND} = E(t+1) \text{ -------- } (8)$$

Optimal timing selection with Nash Equilibrium is provided below:

1) In time$'t,'$ with $c_d < P_b E(t+1)$ and $g'(t) \le 0$, attacker stops the attack, which the administrator acts as a defender to protect.

2) While in $c_d > P_b E(t+1)$ and $g'(t) \le 0$, attacker stops the attack, which the administrator acts as a defender will not protect.

3) The above condition is because;$P_a > P_b$, $g'(t) \le 0$; the finest choice of attackers is not to make any attack whenever defending is considered.

4) The probability of recognizing vulnerability is made by expanding defending cost. Thereby defender will defend them.

5) The administrator will not defend whenever the cost goes higher, $c_d > P_b E(t+1)$.

6) With time $'t', g'(t) > (P_a - P_b)E(t+1), C_d < p_a E(t+1)$ 0-day shows vulnerability and administrator will protect.

7) When $g'(t) > (P_a - P_b)E(t+1), C_d \ge p_a E(t+1)$ 0-day shows vulnerability, and the administrator will not protect.

From the above Nash conditions, attack profit is higher than loss encountered during attack identification. Therefore, the attacker's choice is to attack when defending choice is not encountered.

3.14 Security after Nash Computation

After Nash equilibrium computation for all games, outcomes are measured as a safety measure for unknown threats. With $'t'$ time, jth node of the target shows a negative vector with $P_j(t) = [p_{j,1}(t), \dots, P_{j,k}(t)]^T$is probability distribution of unknown vulnerabilities. $P_{j,m}(t)$ depicts the probability of mth resource used by attackers, where $P_{j,m}(t) \in [0,1], m = 1, \dots, K_j$. Therefore, it is computed that attackers influence node j at time $'t'$ as in Eq. (9):

$$P_j(t) = 1 - \prod_{n=1}^{k_j} \left(1 - P_{j,n}(t)\right) \text{ ------ } (9)$$

Consider that, $P(t) = [P_1(t), P_2(t), .., P_Q(t)]^T$probability distribution of attack nodes at target; where weight is depicted as $W(t) = [w_1(t), w_2(t), \dots, w_q(t)]$. $S(t)$is outcome of security measure as in Eq. (10) & Eq. (11):

$$S(t) = P_j(t).W(t) \text{ ------ } (10)$$

$$S(t) = \sum_{n=1}^{q} P_n(t).w_n(t). \text{ ------ } (11)$$

The computed Equation shows that when the$S(t)$ value is higher, the chances of threat level are also higher.

Algorithm 3: *NASH COMPUTATION*

```
Inputs:
        T: Time step
        Q: Number of attack nodes
```

K_j: Number of resources used by attacker node j

P_(j,n)(t): Probability of the n-th resource used by attacker node j at time t, where n = 1,...,K_j

w_n(t): Weight associated with the n-th attack node at time t, where n = 1,...,Q

Based on these inputs, we can compute the following:

Algorithm:

1. Initialize S(t) = 0.
2. For each attack node n = 1 to Q, do steps 3-4.
3. Compute P_n(t) using Equation (9): P_n(t) = 1 - ∏(1 - P_(n,m)(t)), for m = 1 to K_n.
4. Compute S(t) using Equation (11): S(t) = S(t) + P_n(t) * w_n(t), for n = 1 to Q.
5. Output the value of S(t) as the outcome of the security measure.

Output:

S(t): Outcome of the security measure at time t, representing the threat level. A higher value of S(t) indicates a higher chance of threat.

4. Analysis

The anticipated approach is computed by using numerical simulations with two factors. The first portion of the formula measures traffic volume, root-mean-squared error, and bandwidth computation; the second half of the formula analyses a network by taking zero-day assaults into account using threshold computation.

4.1 Support Vector Machine

The SVM is a kind of supervised machine learning. By a wide margin, the most popular technique of categorization is SVM. Separation between classes is represented by a hyperplane in SVM. In three dimensions, it might produce a hyperplane or group of hyperplanes. Classification and regression analyses benefit greatly from this hyperplane. Objects that aren't backed up by data may nonetheless be classified by SVM based on their attributes. To accomplish segmentation, a hyperplane finds the nearest training site for each class.

Let D = {(xi, yi)} n i=1 be a classification dataset, with n points in a d-dimensional space. Additionally, let us assume that there are only two class labels, Yi ∈ {+1,−1}, denoting the positive and negative classes.

A hyperplane in d dimensions is given as the set of every one point x ∈ R_d that satisfy the equation h(x)= 0, where h(x) is the hyperplane function, defined as follows:

$$h(x) = w^t x + b \text{ --------- (12)}$$

Where w denotes a d-dimensional weight vector and b is a scalar value denoted by the term bias. This research has a hyperplane for points that lie on it.

$$h(x) = w^t x + b = 0 \text{ ----------------- (13)}$$

The hyperplane is thus defined as the set of all points such that$w^t x + b = 0$. To see the role played by b, presumptuous that w1 ≠ 0, and setting xi = 0 for all I> 1, this research can obtain the offset where the hyperplane intersects the first axis, as, by Eq. (14)

$$w_1 x_1 = -b or x_1 = -b/w1 \text{ --------------- (14)}$$

Observe that the support vectors now comprise all points that are on the margin, which have zero slack (ξi = 0) and all points with positive slack (ξi> 0).

Algorithm 4: SUPPORT VECTOR MACHINE

Inputs:

1. D: Dataset for classifying objects, with n points in d dimensions
2. xi: Feature vector of the ith data point
3. yi: Class label of the ith data point, where yi∈ {+1, -1}
4. w: Weight vector of dimension d
5. b: Bias term

Based on these inputs, we can compute the following:

Algorithm:

1. Initialize an empty set, support vectors = {}
2. For each data point i = 1 to n, do steps 3-4.
3. Compute the hyperplane function h(x) using Equation (12): h(xi) = w^T xi + b
4. Check if the data point xi is a support vector by evaluating the slack variable ξi:
   - If ξi = 0, add xi to support_vectors set.
   - If ξi> 0, xi is already considered a support vector.
5. Output the support_vectors set, which contains all the points on the margin (ξi = 0) and points with positive slack (ξi> 0).

Output:

support_vectors: Set of support vectors, which includes points on the margin and points with positive slack.

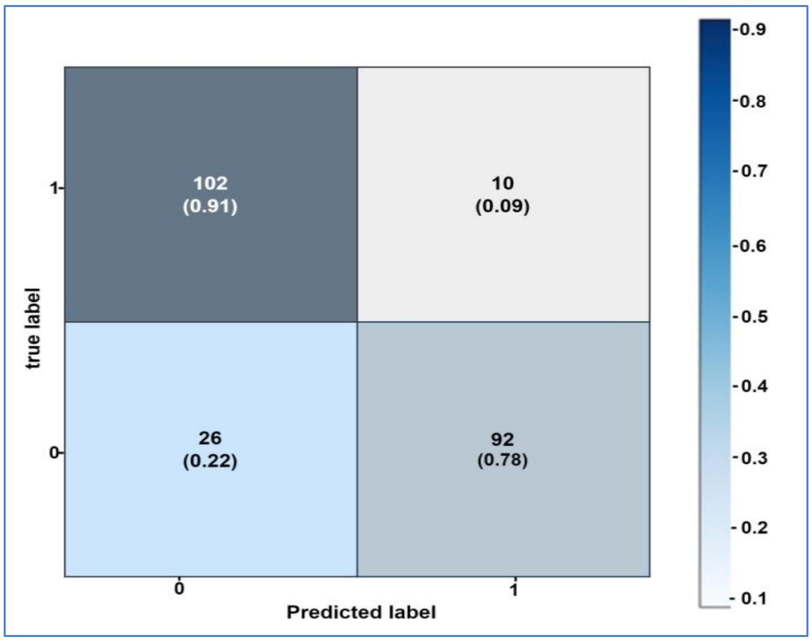

Fig 3. Confusion Matrix for SVM

The confusion matrix is TP, FP, TN, and FN values are represented in Figure 3. The predicted class for TP is 102, TN is 92, and FP and FN are 10 and 26.

4.2 Decision Tree

A DT is an important technique for categorization. It is a method of supervised learning. When the response variable is categorical, a decision tree is utilized. The decision tree model is a tree-like structure showing input characteristics' categorization process. Graphs, text and discrete or continuous data may all be used as input variables.

4.2.1. Steps for Decision Tree Algorithm

1. Create a tree including nodes as input features.
2. Choose a feature to forecast the output from the input feature of zero day attack with the best information gain.
3. For each characteristic in each tree node, the greatest information gain is determined.
4. Repeat step 2 to create a sub-tree that takes advantage of the characters not utilized in the preceding node.

This algorithm takes characteristics with a class of data as input. It employs Decision Tree Classification to construct a collection of rule sets for directly classifying differentiable data. The main benefit of this Classification Algorithm is its ability to recognize and comprehend unmistakably distinguishable properties in a dataset.

The main problem is that it cannot distinguish associated attribute values in the dataset. Some data values may not be classified using any of the decision statements. As a result, such data may be overlooked for research purposes. As a result, it is inadequate for complicated data that combines two components inside the same variable, referred to as multi-variate dataset properties.

The formed results of a decision tree algorithm are based on If. Else rules:

```
IF (Variable (Index) >Threshold limit)
Accept the Positive Result and increment it
Else
Accept the negative result and increment it
```

A looping statement can manage process recurrence, particularly in the research domain for a statement, since it can readily handle multi-dimensional prediction features.

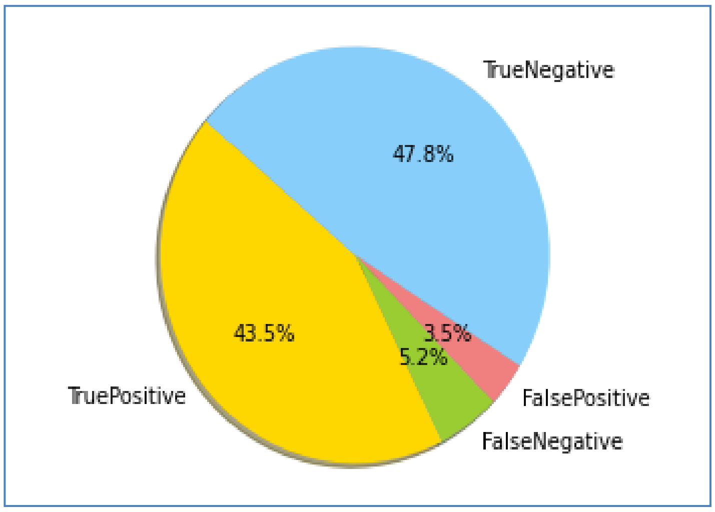

Fig 4. Confusion Matrix for Decision Tree

The confusion matrix is TP, FP, TN, and FN values are represented in Figure 4. The predicted class for TP is 43.5, TN is 47.8, and FP and FN are 3.5 and 5.2.

4.3 G-Naive Bayes (NB) Classifier

An NB classifier is a fundamental probabilistic classifier that relies on Bayes' theorem and independent Bayesian statistics assumptions. Presumably, the existence or absence of a certain class attribute is independent of any other characteristic. Depending on the structure of the probability model, this classifier may be quickly learnt in a supervised learning setting. The Bayes classifier can estimate the classification variables' parameters, such as means and variances, using less training data.

The probability model is conditional, p(C/F1..Fn), on a dependent class variable C with a limited number of outcomes or classes, and is dependent on a large number of characteristics F1 through Fn. A probability table-based model becomes infeasible when the number of features is huge, or a single feature has many potential values. In these cases, the model may be revised following the Equation's specifications

$$p\left(\frac{c}{f1}\dots fn\right) = \frac{p(c)p(f1\dots fn/c)}{p(c)(f1\dots fn)} \text{------------ (15)}$$

It can be simplified as

$$posterior = \frac{priorxlikelihood}{evidance} \text{--------------- (16)}$$

The above statement is the Bayes theorem, and classification is performed using the preceding formula.

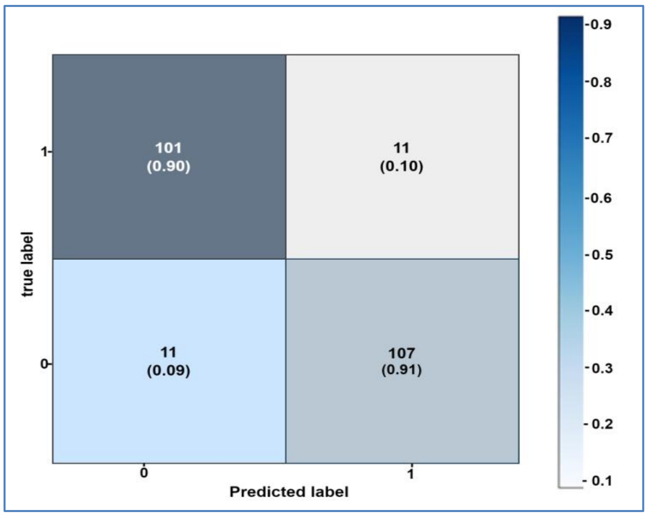

Fig 5. Confusion Matrix for G-Naive Bayes (NB) Classifier

The confusion matrix is TP, FP, TN, and FN values are represented in Figure 5. The predicted class for TP is 101, TN is 107, and FP and FN are 11 and 11.

4.4 Logistic Regression Algorithm

The LR represents the identification of relationships based on interrelated data values. The regression process entails identifying overlapping data values to create a new data set. The random forest tree is one such technique that identifies the association between data attributes and outcomes. The approach is a meta-estimator that may be expanded to create data trees without having to make a judgment about separable data.

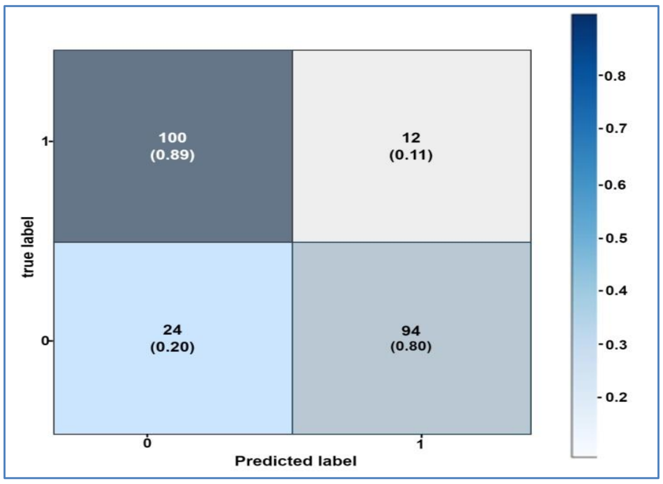

Fig 6. Confusion Matrix for Logistic Regression

The confusion matrix is TP, FP, TN, and FN values are represented in Figure 6. The predicted class for TP is 100, TN is 94, and FP and FN are 12 and 24.

4.5 Stacking Ensemble Classification

Stacking Ensemble Classification is a machine learning technique that combines multiple individual classifiers to create a powerful predictive model. In this approach, several base classifiers are trained on the same dataset, and their predictions are used as input features for a meta-classifier, which makes the final prediction.

The numerical analysis recommends that the anticipated model effectively deals with a 0-day attack environment with minimal differences observed for providing alert message latency. It is proven from the obtained graph that the anticipated model can deal with a network of zero-day attacks with effectual variation among values, as shown in Table II. From these outcomes, values are noted for examining message latency for data sharing protocol with Eq. given above. When message latency is lesser, infected nodes do not show a vast infected environment. Message latency may also increase when the size of the network is also larger. The overall difference between the present scenario and the observed value is lesser; therefore anticipated model can be utilized for reliable communication during a 0-day attack.

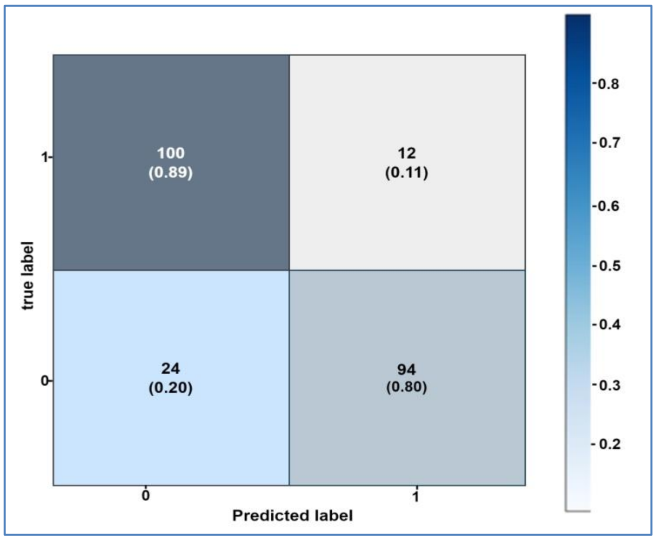

Fig 7. Confusion Matrix for Stacking Ensemble

The confusion matrix is TP, FP, TN, and FN values are represented in Figure 7. The predicted class for TP is 100, TN is 94, and FP and FN are 12 and 24.

4.6 Analysis Interpretation

The CloudSim simulator is used to conduct these analyses in a network that has been subjected to numerical analysis. Connectivity among the clustered network is often supported by a network with a base station. The network is made up of stable parts, so even a zero-day exploit may not compromise the whole system. This model examines the relationship between network size and features like dependability and consensus. Clustered networks may be linked to the overall network size, which is calculated as the total of all the network's active nodes. The original network's size is calculated after zero-day-vulnerable nodes are isolated from the system using the forecasted model. The detection threshold and the mean squared error could both improve if the threat of zero-day attacks is mitigated. As a calculating cutoff, let's choose 0.5. Below this number, communication is not deemed trustworthy; above this value, the network is secure even if some nodes are infected. Reliability in networks is improved by eliminating nodes infected with malware. Due to the lack of expected forward mechanisms, some traffic rates will rise if certain nodes remain in the network. This causes more labour and lowers network dependability. A network that eradicates these zero infectious nodes may possess a better cost that is lesser than a network with infected nodes. Infected nodes in the network possess higher costs, as actual network cost computation includes infected nodes. Based on infection identity, non-transmission packets are provided to infected nodes. These are done with the anticipated model. Here, two diverse protocols have been used for alert messaging and sharing data in times of crucial network instances. Latency analysis is done with these protocols and assists in understanding communication overhead caused owing to messages among network entities. Diverse metrics like traffic rate, error rate are considered for both infected and typical scenarios. While analyzing a network, network size comprises nodes involved in communication among clustered nodes. Numerical analysis values are utilized by sensing various iterations needed for producing a particular message. It's also possible for these figures to change based on how the nodes' communication protocol is set up. Nodes in a network may be identified during a 0 day assault by using critical instances. Eq. (17) is used to provide notifications for certain messaging protocols:

$$L = 2T + 3T + 4T + 2T + T \text{ ----- (17)}$$

The final alert message produces latency after decision making is evaluated as in Eq. (18):

$$L = L_i + 4Q + Q + 3Q \text{ ------ (18)}$$

Moreover, network with equal decision-making delay with Eq, (19):

$$L = L_i + 8Q \text{ ------ (19)}$$

At present, in a linked state, communication among clustered nodes produces packet delay as in Eq. (20):

$$D(P, E) = \frac{P*E}{\beta} + D_{wired} \text{ ------- (20)}$$

In wireless links, packet delay is provided as in Eq. (21):

$$D(P) = F + (x - 1)t \text{ ------- (21)}$$

Then, communication overhead for alerting messaging as in Eq. (22):

$$C_{overhead} = 2D(P) + 2D(P) + 3D(P, E) + D(P, E) + 4D(P, E) + L \text{ ------- (22)}$$

In the next stage, analysis is carried out in data sharing protocols, generating message latency is modeled as in Eq. (23):

$$L = 3T + 3T + 2T + 4T + 5T \text{ ------- (23)}$$

Where final critical message production latency is provided as in Eq. (24):

$$L = L_i + 5\,Q \text{ ------- (24)}$$

At last, the communication overhead for sharing data is computed as in Eq. (25):

$$C_{overhead} = 3D(P) + 5D(P) + 3D(P, E) + 2D(P, E) + 5D(P, E) + L. \text{ --------- (25)}$$

5. Experimental Results and Discussion

Our result was analyzed using the framework of many classifications. The multiclassification data were partitioned into two classification issues, and the one-versus-the-rest approach was applied.

TPd: the prediction is category d, and the reality is category d.

TNd: the prediction is other classes of category D; the reality is other classes of category d.

FPd: the prediction is category D; the reality is other classes of category d.

FNd: the prediction is other classes of category D; the reality is category d.

Each category was used as a positive sample to determine the overall accuracy, precision, and recall values. Equation (26) may be used to represent the precision as follows:

$$\text{Accuracy} = \frac{\text{Numberofsamplescorrectlyclassified}}{\text{Numeberofsamplesforallcategories}} \text{ -- (26)}$$

As demonstrated in Equation (27), the precision of a given category may be used as a predictor of the reliability of the sample as a whole:

$$\mathrm{Precision_i} = \frac{\mathrm{TP_d}}{\mathrm{TP_d + FP_d}} \text{ ------ } (27)$$

As indicated in Equation (28), recall may be thought of as the percentage by which a correctly predicted sample of category d covers the percentage of category d in the sample set,

$$Recall_i = \frac{TP_d}{TP_d + FN_d} \text{ -------- } (28)$$

F measure is calculated by giving the Equation.

$$F - Measure = 2.\frac{Precison.recall}{Nprecison+recall} \text{ ------- } (29)$$

Table 1 displays the results of training and testing the suggested model on the dataset used for this investigation. Modified Bi-LSTM Deep Neural Networks have been used in training and testing.

Table 2. Training and testing values with the 10 Epoch

Epoch	Training Loss	Validation Loss	Training Accuracy	Testing Accuracy
1	00.1552	00.0614	00.9533	00.9849
2	00.0531	00.0493	00.9843	00.9840
3	00.0341	00.0474	00.9893	00.9849
4	00.0224	00.0455	00.9931	00.9846
5	00.0147	00.0589	00.9954	00.9827
6	00.0110	00.0521	00.9963	00.9850
7	00.0082	00.0510	00.9972	00.9859
8	00.0061	00.0533	00.9979	00.9869
9	00.0047	00.0608	00.9985	00.9852
10	00.0053	00.0533	00.9982	00.9872

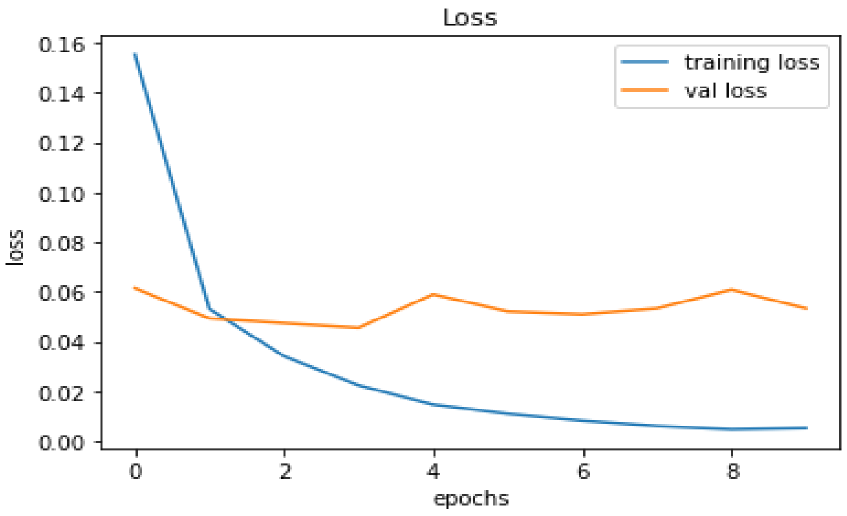

Fig 8. Training and Testing Loss

The proposed model is trained with loss values, as shown in Figure 8. In X-axis denotes the Epoch number, and Y-axis denotes the loss value.

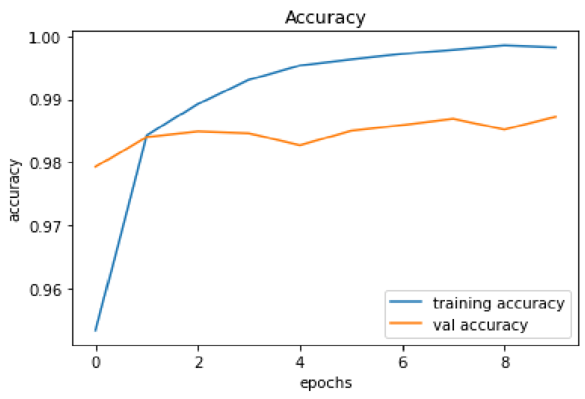

Fig 9. Training and Testing Accuracy

Figure 9 displays the accuracy of the CNN-ResNet's tests after it was trained for 10 iterations using Training. The Epoch number (x-axis) and precision (y-axis) are shown against one another. Based on the CSV file, it will get the accuracy and f-measure values using ML algorithms.

Table 3. Al

Dataset	Algorithm	Accuracy	Precision	Recall	F-measure
Dataset 1	DT	90	91	88	90
	SVM	84	80	90	85
	GNB	90	90	90	90
	LR	84	81	89	85
	Stacking Ensemble Classifier	95.4	89.3	91.5	90.4
Dataset 2	DT	90	87	94	90
	SVM	85	81	91	86
	GNB	90	90	90	90
	LR	84	81	89	85
	Stacking Ensemble	95	88.3	89.8	89.1

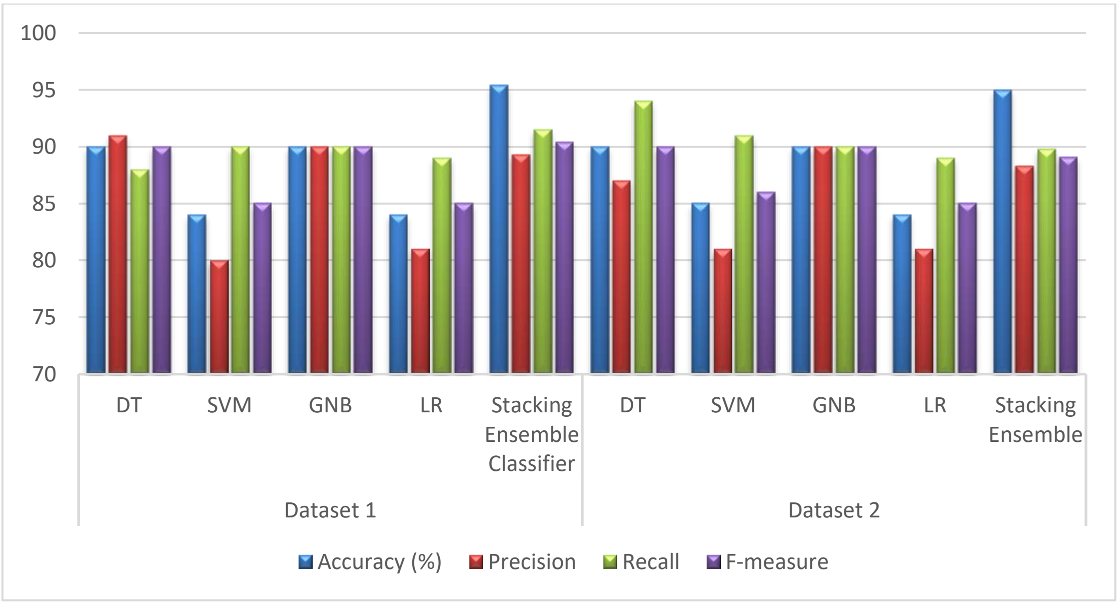

Fig 10. ML Algorithm Comparison Chart for Accuracy and F-Measure Values based on Different Datasets

The CSV dataset has classified LR, DT, SVM and GNB algorithms, as shown in Figure 10 and figure 11. In X-axis denote the ML algorithms, and Y-axis denotes the accuracy. The values are represented in table 3.

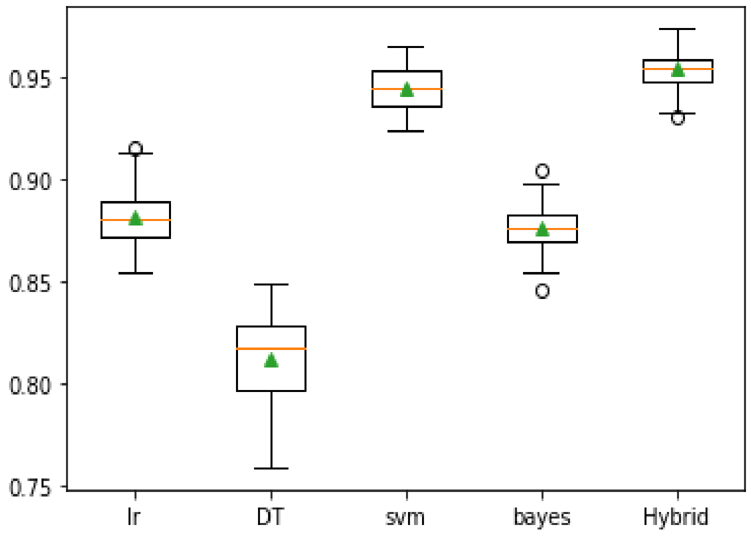

Fig 11. Performance Evaluation

The ultimate cause of disagreements relies upon complex quantifying real-world vulnerability impact. Disclosure and releases patches devoid of comprehensive field data. Here, the initial step to achieve this goal is by demonstrating 0-day vulnerability that causes significant end-users risk as attacks volume increases by magnitude. Moreover, vendors may prioritize patch vulnerabilities, providing more efficiency to vulnerabilities to be disclosed. For instance, 80% of vulnerabilities are discovered more than three days before the disclosure date. However, after patching become, users frequently delay its deployment. For example, typical windows users may manage 14 mechanisms to maintain the host wholly patched.

Similarly, evidence suggests that attackers adopt strategies to expect zero-day vulnerability disclosure. With raised suspicion, attackers did not exploit until the deployment of the patch was imminent. Exploit files utilized in an attack against RSA were transmitted to various organizations for vulnerabilities disclosure. This is due to earlier disclosure of reduction with zero-day vulnerabilities. For instance, new exploits are paid in installments, with every payment based on patch lack. This research is essential for quantifying complete disclosure factors and measuring how quickly vulnerable hosts are patched to vulnerability disclosures.

6. Conclusion

Identifying and preventing zero-day attacks and vulnerabilities is incredibly important in various networking setups. Using the network defense perimeter as an attack vector, these attacks can inflict damage on various devices. Gaming theory with Modified Bi-LSTM is used to conduct a deep examination into the 0-day attack and preventive measures. This work predicts Nash equilibrium for zero-day attack mitigation in networking. Using the behavior of the networking system as a detection tool, message protocol and data exchange methods, and gaming theory interpretation, the predicted strategy would provide reliable communication while also preventing or mitigating network attacks. The predicted protocol was calculated numerically, and the results suggest that zero-day attacks may be eliminated in the network without affecting performance. The current comparison decides the predicted model's performance and benchmark standards, which justifies the performance and benchmark standards. The bandwidth and mean error rate results indicate that the system will perform better during zero-day attacks. Concentrate efforts in the future on securing the passes of the message protocol and the data sharing protocol by inspecting them using various security methods. Furthermore, the anticipated model will be investigated by analyzing the size of messages and the actual behaviors of various devices.

References

[1] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan, “Fast portscan detection using sequential hypothesis testing,” in Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on, May 2004, pp. 211–225.

[2] G. R. Hendry and S. J. Yang, "Intrusion signature creation via clustering anomalies," in Proc. of SPIE 2008, pp. 69730C-1.

[3] J. Song, H. Ohba, H. Takakura, Y. Okabe, K. Ohira, and Y. Kwon, "A comprehensive approach to detect unknown attacks via intrusion detection alerts," presented at the Proceedings of the 12th Asian computing science conference on Advances

in computer science: computer and network security, Doha, Qatar, 2007.

[4] AlEroud and G. Karabatis, "Discovering Unknown Cyber Attacks using Contextual Misuse and Anomaly Detection " ASE Science Journal vol. 1, pp. 106-120, 2012.

[5] AlEroud and G. Karabatis, "A Contextual Anomaly Detection Approach to Discover Zero-Day Attacks," in 2012 ASE International Conference on Cyber Security, Washington, D.C., USA, 2012.

[6] D. M. J. Tax and R. P. W. Duin, "Data description in subspaces," in Proceedings. 15th International Conference on Pattern Recognition, pp. 672-675 vol.2.

[7] X. B. Li, "A scalable decision tree system and its application in pattern recognition and intrusion detection," Decision Support Systems, vol. 41, pp. 112-130, 2005.

[8] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier," ExpertSystems with Applications, vol. 39, pp. 13492-13500, 12/15/ 2012

[9] E. Eskin, A. Arnold, M. Prerau, et al., "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," Applications of Data Mining in computer security, vol. 6, pp. 77-102, 2002.

[10] S. A. Zonouz, R. Berthier, H. Khurana, W. H. Sanders, and T. Yardley,"Seclius: an information flow-based, consequence-centric securitymetric," Parallel and Distributed Systems, IEEE Transactions on, vol.26, pp. 562-573, 2015.

[11] D. M. Lewis, and V. P. Janeja, "An empirical evaluation of similarity coefficients for binary valued data," IGI Global,2011, pp. 44-66.

[12] P. Ning, Y. Cui, D. S. Reeves, et al., "Techniques and tools for analyzing intrusion alerts," ACM Trans. Inf. Syst. Secur., vol. 7, no. 2, pp. 274-318, 2004.

[13] J. Mchugh "Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations performed by Lincoln Laboratory," ACM Trans.Inf. Syst. Secur., vol. 3, no. 4, pp. 262-294, 2000.

[14] M. Wu, and C. Jermaine, "Outlier detection by sampling with accuracy guarantees," In Proc. of the 12th ACM SIGKDD Int'l conf. Knowledge discovery and data mining, Philadelphia, PA, USA, 2006, pp. 767-772.

[15] K. Zhang, X. Liang, R. Lu, and X. Shen, “Sybil attacks and their defenses in the internet of things,” IEEE Internet of Things Journal, vol. 1, no. 5, pp. 372–383, 2014.

[16] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and privacy challenges in industrial internet of things,” in Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference (DAC ’15), pp. 1–6, IEEE, San Francisco, Calif, USA, June 2015.

[17] Kotenko and A. Chechulin, “Attack modeling and security evaluation in Siem systems,” International Transactions on Systems Science and Applications, vol. 8, pp. 129–147, 2012.

[18] F. Kamm¨uller, M. Kerber, and C. W. Probst, “Insider threats and auctions: Formalization, mechanized proof, and code generation,” Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 8, no. 1, pp. 44–78, 2017.

[19] L. Bilge and T. Dumitras, "Before we knew it: an empirical study ofzero-day attacks in the real world," in Proceedings of the 2012 ACM conference on Computer and communications security, 2012

[20] G. Bonfante, M. Kaczmarek, and J.-Y. Marion, “Morphological detection of malware,” in Proceedings of the 3rd International Conference on Malicious and Unwanted Software, MALWARE2008, pp. 1–8, USA, October 2008.

[21] Santos, F. Brezo, J. Nieves, et al., "Idea: Opcode-sequence based malware detection,” Lecture Notes in Computer Science(including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface, vol. 5965, pp. 35–43,2010.

[22] Niki, “Drive-by download attacks: Effects and detection methods," in Proceedings of the 3rd IT Security Conference for the Next Generation, 2009.

[23] E. Al Daoud, I. H. Jebril, and B. Zaqaibeh, “Computer virus strategies and detection methods,” International Journal of Open Problems in Computer Science and Mathematics, vol. 1, no. 2, pp.12–20, 2008.

[24] N. Nissim, R. Moskovitch, L. Rokach, and Y. Elovici, “Novel active learning methods for enhanced PC malware detection in windows OS,” Expert Systems with Applications, vol. 41, no. 13,pp. 5843–5857, 2014

[25] Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as a representation of executables for data mining-based unknown malware detection,” Information Sciences,vol. 231, pp. 64–82, 2013.

[26] M. Alazab, S. Venkatraman, P. Watters, and M. Alazab, "Zero-day malware detection based on supervised learning algorithms of API call signatures," in Proceedings of the Ninth

Australasian Data Mining Conference-Volume 121, 2011, pp. 171-182.

[27] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: scalable and accurate zero-day android malware detection," in Proceedings of the 10th international conference on Mobile systems, applications, and services, 2012, pp. 281-294.

[28] Y. Park, D. S. Reeves, and M. Stamp, “Deriving common malware behavior through graph clustering,” Computers &Security, vol. 39, pp. 419–430, 2013.

[29] Z. Chen, M. Roussopoulos, Z. Liang, Y. Zhang, Z. Chen, and A. Delis, “Malware characteristics and threats on the internet ecosystem,” The Journal of Systems and Software, vol. 85, no. 7,pp. 1650–1672, 2012.

[30] X. M. Choo, K. L. Chiew, D. H. A. Ibrahim, N. Musa, S. N. Sze, and W. K. Tiong, “Feature-based phishing detection technique,” Journal of Theoretical and Applied Information Technology, vol.91, no. 1, pp. 101–106, 2016.

[31] Khoury, J., &Nassar, M. (2020). A Hybrid Game Theory and Reinforcement Learning Approach for Cyber-Physical Systems Security. NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium. doi:10.1109/noms47738.2020.9110453

[32] Stier, J., Gianini, G., Granitzer, M., & Ziegler, K. (2018). Analysing Neural Network Topologies: a Game Theoretic Approach. Procedia Computer Science, 126, 234–243. doi:10.1016/j.procs.2018.07.257

[33] Xu, J., Alsabbagh, A., & Ma, C. (2022). Prediction-Based Game-Theoretic Strategy for Energy Management of Hybrid Electric Vehicles. IEEE Journal of Emerging and Selected Topics in Industrial Electronics, 3(1), 79–89. doi:10.1109/jestie.2021.3087962

[34] Gaao, L., Li, Y., Zhang, L., Lin, F., & Ma, M. (2019). Research on Detection and Defense Mechanisms of DoS Attacks Based on BP Neural Network and Game Theory. IEEE Access, 7, 43018–43030. doi:10.1109/access.2019.2905812

[35] Swathy Akshaya, M., and G. Padmavathi. "Zero-Day Attack Path Identification using Probabilistic and Graph Approach based Back Propagation Neural Network in Cloud." *Mathematical Statistician and Engineering Applications* 71.3s2 (2022):1091-1106