CHAPTER - 5

ZERO-DAY ATTACK PATH IDENTIFICATION USING PROBABILISTIC AND GRAPH APPROACH-BASED BACK PROPAGATION NEURAL NETWORK IN CLOUD

5.1 Introduction

This chapter introduced EBPNN model to find zero-day attack paths. This outlined the model structure, learning process and reason why it works much better than other methodologies based on Bayesian. The addition is incorporation of adaptive learning rate and momentum in combination with the classic BPNN to achieve enhanced convergence and pattern recognition. EBPNN was best improvement of up to 3.01% of better accuracy and showed general improvement in all measurements of evaluation compared to existing approaches. The next chapter presents a zero-day vulnerability predictive model that is informed with combination of Hybrid Game Theory and Neural Networks and attacker-defender dynamics. The model is a combination of EBPNN and active anticipation of threats prior to spread.

The adventures of zero-day attacks will typically entail a serious chain of activities to form what is termed as a zero-day attack path. The significance of identifying such paths is that it provides immediate intrusion detection, flow of attack process analysis and proactive protection initiation before the execution of massive damage. Comparing to the systems that concentrate on vulnerability of the person, attack path detection provides the general system image of attacker, which is vital to pre-empt the further actions and prevent further use of the system.

This chapter presents a creative, probabilistic, and graph model approach for identification, detection, and prediction of the zero-day attack paths using Enhanced Back Propagation Neural Networks (EBPNN). The proposed methodology models evasive behavior of attackers through dynamic graph representations, enabling systematic analysis and visualization of the complete attack lifecycle. The model is trained with structural benefits of graph representations along with the learning capabilities of neural networks to identify covert behaviors and potential attacks in real time, without relying on predefined

attack signatures. This approach is particularly effective in modern cloud environments, where attack surfaces are highly dynamic and conventional defense mechanisms often prove insufficient.

It is an original piece of work as it combines graph-amplified learning with inference neural-based zero-day detection. The context relationship between attack steps and BPNN used makes the use of instance graphs to facilitate adaptive learning based on data of system behavior which is complex and high-dimensional. Such an approach in contrast to other research works is flexible in real-time and proactive in protection hence power of prediction and reaction to dynamic attacks is enhanced significantly. The significant works of the stage include: (1) a more effective BPNN based approach of path discovery of zero-day attacks, (2) dynamic generation of instance graphs that can give information of the complex multi-stage attacks, (3) experimental results of the superior detection over the baseline methods, and (4) a smart and scalable defense system which could predict the progress of attacks.

In the chapter, the author discusses the zero-day attack path discovery by using Back Propagation NN. This methodology is explained in Section 5.5, the findings of the experiment and the discussions are provided in 5.6 and summary of this chapter is provided in 5.7.

5.2 Dataset Justification and Simulation Approach

Zero-day attacks are never predictable and it takes place in an unknown way. Thus, it lacks numerous and publicly available datasets of real-life attacks as a zero-day. The majority of benchmark datasets applied to cybersecurity such as KDD99, NSL-KDD and CICIDS2017, contain the information about already known attacks and are not able to provide the insight into the dynamism/adversarial part of zero-day threat. Moreover, these datasets are often characterized as the imbalance of classes, incorrect protocols, and insufficient richness of behaviors to model proactive attacks.

To remove these limitations, first foundation of this research will be a simulation-based approach with the assistance of CloudSim that is a well-known toolkit to simulate and model cloud computing systems. The pseudo-data is be utilized to create simulated attack behavior and in controlled conditions under which basic attributes of a system such as

interaction of the nodes, exploitation of resources and the time variations are considered. This way allows us to generate a zero-day-like situation with no a-priori known attack signatures and this makes model more robust and general.

The obtained data have been developed to support all four stages of the proposed model, i.e., finding attack route, predicting behavior of the attackers, identifying whether or not adversarial patterns exist, and optimizing the classification. It contains realistic system interfaces that are required to train and test Enhanced BPNN, Bi-LSTM using Game Theory, DC-nZDA and OLFFOA models. In this case, simulation as an effective need because of inaccessibility of the data sets is not simply a strategic decision, but also an effort to research the system, in detail and under the controlled, yet realistic attack environment.

In this research, Phase 1 utilizes the simulated dataset generated through CloudSim to model zero-day attack paths. Although the dataset is synthetic, it is carefully designed to mimic real-time cloud environments by incorporating dynamic network conditions, varying workloads, and multiple attack scenarios.

The proposed Enhanced BPNN model is inherently generalizable, as it is based on adaptive learning mechanisms that can handle diverse input patterns and is not restricted to a specific dataset or environment. This enables its applicability across different cloud systems, network environments, and real-world scenarios.

Real-time effectiveness is achieved as the proposed framework extends beyond static simulation. The outputs from Phase 1 are processed in Phase 2 and Phase 3 using advanced machine learning and deep learning models such as Bi-LSTM and DCNN, which are capable of handling continuous, high-volume, and dynamic data streams.

To handle bursty traffic conditions, the system uses the temporal learning capability of Bi-LSTM to capture sudden spikes and variations in traffic patterns. Additionally, DCNN integrated with ResNet enables fast and accurate real-time attack detection. These models are well-suited for managing high-speed and irregular traffic behavior.

Furthermore, Phase 4 optimization using Fruit Fly Optimization with Levy Flight enhances system scalability, reduces computational latency, and improves response time, making the framework suitable for real-time deployment.

Therefore, although the dataset is generated through simulation, the proposed multi-phase architecture, adaptive learning mechanisms, scalability, and low-latency response ensure that the framework can be effectively extended to real-time cloud environments with bursty traffic and rapid response requirements.

5.3 Key Defense Measures

This section portrays the key defense measures that are used in the research.

a. Real-Time Adaptability

Real time adaptability is the capability of a system to respond to emerging new threats and is in real-time and does not need assistance of preset signatures. This is a condition that is required whenever it is involved in dealing with zero-day attacks. The attacks of zero-days can be attributed to exploitation of unknown weaknesses and, therefore, the methodology of scanning, which is grounded in the application of signature, is not beneficial. The introduction of EBPNN and graph-based modeling to adjust towards unknown threat and implementation of new behaviors into system will make it more adaptive. It means that threats emerging can be updated continuously, and responses are prepared to detect threats and neutralize them as soon as possible and therefore more resilient based on the cumulative resistance ability.

b. Proactive Defense

Proactive defense is a preventive control that proactively forecasts potential attacks with reference to behavior of system and forecasts the movement of adversaries before it can inflict any harm to the system. Unlike the old methodologies or defenses that were signature and patch based, Proactive Defense Framework can apply Game Theory and ML models to simulate the decisions made by attackers and then devise an action strategy and ultimately predict what attack vectors, attackers might have. Yet, through prioritization and prioritization of problems so that it cannot be sabotaged, to prevent destruction of the security of issues, general risk is minimal and security state of any organization attacked is adaptive, and lastly, an enhanced method of enriching substantive defenses against sophisticated cyber-based attackers.

c. The Defense System of the Enlightenment.

Smart defense architecture exploits the services provided by ML and DL to infer and identify advanced attack patterns that might be used by the intelligent defense architecture

to make informed decisions based on evolving, or learned behavior of attack patterns instead of providing naive and rule-based detection. Overall effect of the old fashioned rules based detection systems is that cannot protect new and emerging attacks that never faced or never yet saw or experienced the attack in rules. The proposed architecture will include the functionalities of EBPNN, ResNet50 and LSTM network which will offer intelligence and autonomy in training and detecting complex zero-day attacks and the detection when the behavioral pattern changes to adjust the detection plan to the new and unique attacks.

d. Robustness

An example of such form is that of robustness which is noted in the ability of system to support a large amount of accuracy in its identification, irrespective of whether the situation is noisy, imbalanced, or antagonistic. It is a result of a cyber attack whose data cannot be more than environmental condition which is not always the entire picture and is active deceiving and at the same time, could not be detected by the traditional models with all the required level of confidence. The ensemble classifier that is being optimized by OLFFOA is what causes the proposed models not to be as affected by change in data, less number of false-positives, and overall less prone to generating high numbers of false negativity in general as more reliable regardless of the dataset in question.

e. Scalability

Scalability is a notion or a capacity of a system to be in a state to handle a larger amount of data, complexity and even the network size without affecting the performance level. In cloud computing whereby huge processes of traffic are being done at any given moment, the classical detection processes cannot keep abreast with the increasing computing requirements. It is capable of providing both detection and analysis capable of scaling well with the assistance of scalable algorithms, such as a graph based EBPNN, and distributed simulations in CloudSim, allowing it to be flexible and adaptive to change under long-term demands and to adapt to dynamically changing environment.

f. Cardinality of Reporting True Cyber Attacks.

The intricacy of real cyberattack information implies the intricacy and multi-tiered state of real attacks, which are generally characterized as one that is multi-stage, dynamic interactions and unseen dependencies of the systems. Real cyberattacks are dynamic at host level, process and time wise and are subtle as opposed to simplified or artificial data. To

address this problem, proposed framework will be a blend of instance graph modeling and DL that allow the framework to learn hidden patterns, association and multi-stages attack path which are not feasible in the static and rule based models. This makes the sophisticated cyber threats more authentic and wholesome.

5.4 Ground Works On Zero-Day Path Identification

The primary works in this Chapter (Path Identification) could be pointed at as possessing origins in three major works concerning the establishment of roots of basic works of zero-day attack paths identification. The PATROL (Dai, Sun, and Liu, 2013) has already been introduced as suspicious propagation paths of intrusion with no vulnerability information, but it contains the problem of path explosion and noise sensitivity due to rule implementation. On the one hand, it was presented as a system object dependency graph (SODG). It has been introduced with ZePro in its original form (Sun et al., 2016) that included BNs to provenance graphs to compute probability of infection and offered probabilistic access to possible zero-day paths. It was furthered in the form of a long journal (Sun et al., 2018) that introduced a more mature ZePro system that would also be capable of aggregating multiple types of intrusion evidence and estimate risk of it resulting in an infection, and prove useful in uncovering multi-step zero-day exploits. All this accumulated to produce the paradigm of provenance-graphs and to make probabilistic inference and not deterministic, rule-based reasoning. The given solution will be based on the same paradigm and will address the issue of scalability, adaptability and nonlinear dependency modeling with an EBPNN-based inference engine that will use temporal, contextual, structural features to learn zero-day paths with a higher level of precision and scale.

5.5 Proposed Methodology

The proposed introduce a new approach of detection, recognition and predicting of zero-day attack paths in chapter using a Probabilistic Graph-Based Enhanced BPNN. The method represents path of attacks as a graph of object occurrences and applies neural learning to precisely determine the paths. The method has been adjusted in order to counter the shortcomings of conventional intrusion detection systems based on recognizable attack signatures.

All hosts monitor system calls independently and capture the resulting traces of execution. These traces are then transported safely to centralized investigation server where are analyzed offline. At this point, a grounded BN is used to build an instance graph, and it is possible to detect and reconstruct the possible attack paths effectively. The general workflow of proposed system is shown in Figure 4.1.

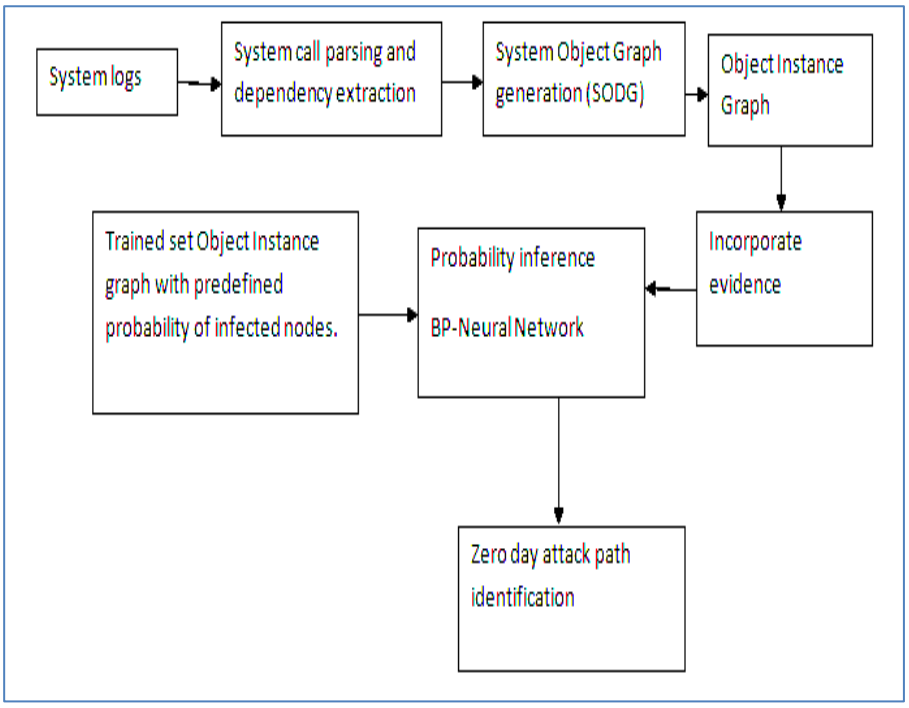

Figure 5.1 Flow Diagrams for the Proposed Work in Phase 1

The application of the Enhanced BPNN in determination of a path towards a zero-day attack is illustrated in figure 5.1. The logs of network traffic are pre-processed and patterns of the attack in form of graphs including host vulnerabilities, dependencies and topology is extracted and inputted into EBPNN which has been trained on known paths simulated by PATH dataset. In the testing step, tests are carried out on invisible samples and the nodes retrieved with an EBPNN probability of greater than 0.8 are assumed to be considered as members of a zero-day attack path. The maximum accuracy, precision, recall as well as F1-score were 99.0, 99.21, 99.34 and 99.54 respectively. The performance of EBPNN is also effective and it is supported with the misclassification rate of 1.0 percent besides it performs better than Bayesian and Probabilistic Bayesian Network models.

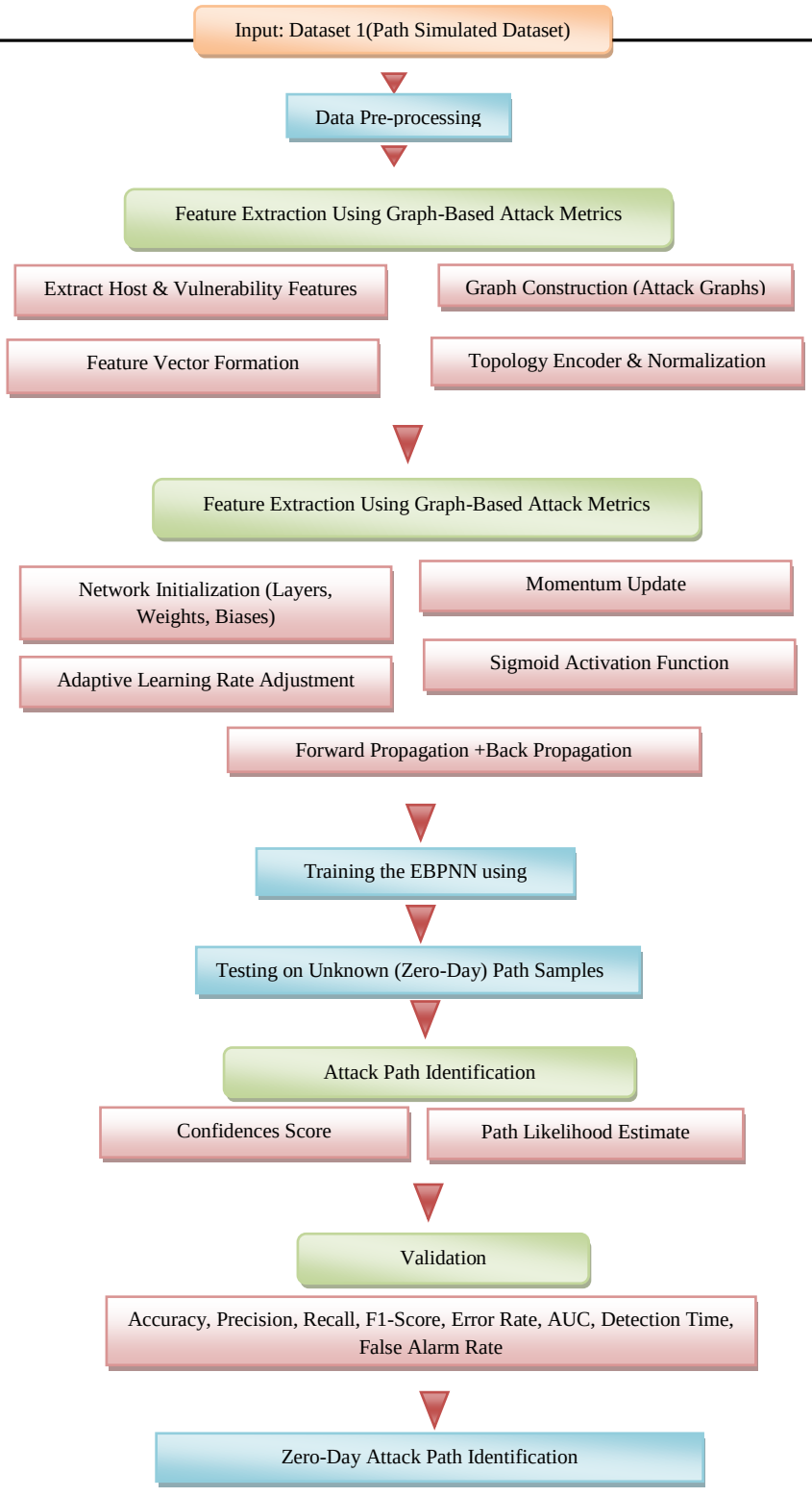

Figure 5.2 Proposed Framework of Zero Day Attack Path Identification in Phase 1

Figure 5.2 shows how zero-day attack could be detected using EBPNN. The model involves first step of preprocessing network traffic records and creating graphic features of attack such as vulnerability of the host, dependency relationship and network topology.

These are elements that are inputted into trained EBPNN that is trained by experimented attack paths and tested by unknown zero-day samples in the past. This is the stage in which standard measures of EBPNN. System performance is determined by standard measures, such as accuracy, precision, recall, and false alarm rate. Particularly, its application to simulated PATH data (D1) yields the accuracy of the model of 99.0, the precision of 99.21, the recall rate of 99.34, and the false alarm rate of 0.46 that is effective zero-day attack path detection with graph enhanced EBPNN.

5.5.1 Algorithms

i. **Logging:** Each host is seen to make system call to all processes that are active. The schedule implementation and interaction is documented. The log file will also manage to indicate communication between two processes or files that had been involved in the process. Examples of data which can be stored in system call auditing are a) socket communications, each instance graph of host, b) Node identifiers and absolute file pathnames derived out of OS knowledge. c) The Timestamp of the calls in the individual system. System call filtering saves unnecessary bandwidth and CPU resources by filtering off redundant or benign objects that would otherwise use system resources.

ii. **Parsing into System Objects:** The operating system object instances and associated dependencies relationships are extracted and interpreted using system calls. These parameters also do exist in the parsing. It adds labels to nodes and uniquely identifies them as well as aiding in drawing conclusions about the direction of the edges. System objects are obtained out of the host files of each and every host.

iii. **Instance Graph:** This is called parsing and transforms the information obtained to nodes and directed edges. An object instance graph is then drawn based on the dependency relationships of the system objects as shown below.

iv. **Construct Dependency Objects:** To model the dependencies and trace spread of intrusion through the system, a super-graph is built by examining the system calls.

v. **System Object Dependency Graph:**

   a. When a trace is collected for a host it is represented as$\sum x$ and modeled as a directed graph G(V_x, E_x) . Here, $V_x$denotes the set of nodes, initially empty{$\emptyset$}

and $E_x$denotes the set of directed edges, also initially empty $\{\emptyset\}$. as the system executes system calls, the corresponding dependency relations are extracted from each$syscall$. Based on predefined dependency rules, the dependency Dep$\in \{(src \leftarrow sink),(src \rightarrow sink), \quad (src \leftrightarrow sink)\}$, where src and $sink$ represent OS objects. The node set is updated as $V_x = V_x U\{src, sink\}$ and the edge set is updated as $E_x = E_x U\{dep\}$.

b. In case (a$\rightarrow b$) $\in E_x$p and (bc)$\in E_x$, then c based transitively on a. Into system dependencies and objects have been parsed by system calls. System calls are processed according to dependency rules. Following such rules, every system call can be subdivided into three components, namely, a source object, a sink object, and a dependency relationship. These objects extracted and dependencies are then combined to form a directed graph as shown in the following table.

Input:

Σ_x : System call traces for the xth host
Dependency rules parsed from syscalls

Output:

G(V_x, E_x) : Directed graph representing the system object dependency graph for the host

Steps:

1. Initialize node set: V_x ← ∅
2. Initialize edge set: E_x ← ∅
3. For each system call trace t in Σ_x do:
   a. Extract src ← source object from t
   b. Extract sink ← sink object from t
   c. Extract dep ← dependency relation from t
   d. V_x ← V_x ∪ {src, sink} # Add nodes
   e. E_x ← E_x ∪ {dep} # Add edge
4. Construct graph G with vertices V_x and edges E_x
5. Return G

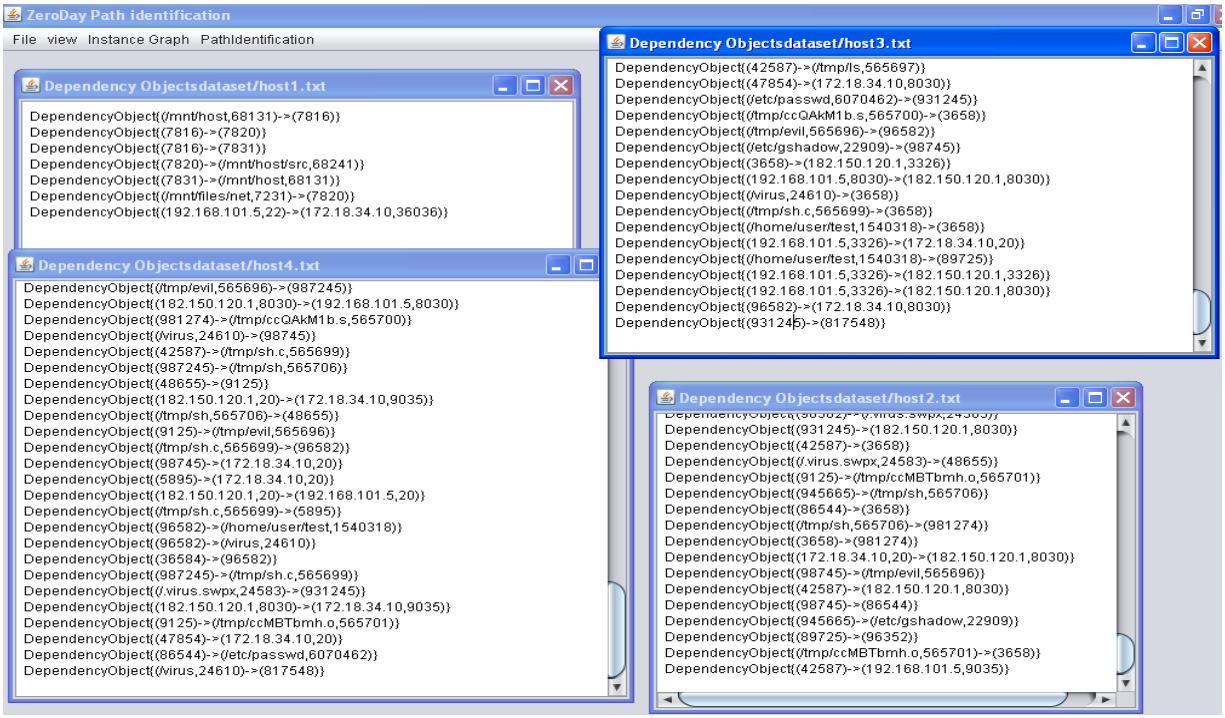

Figure 5.3 Dependency Objects

vi. Construct the Object Instance Graph: The dependency graphs are represented by one of the dependency graphs. Object Dependency Graph A node depicts a node in an object instance at a given time. Moreover, there are also numerous cases in which there are identical instances of the object in the time frame having multiple stages of infection. This size of the graph is equivalent or bigger.

When the system calls trace T $[t_{begin,}t_{end}]$ within a time window is mentioned, thenO_T,object instance graph$G_T(V.E)$is represented by V denoting node-set (initialized to set of 0) and E denoting directed edges set (initialized to set of 0).

a. In the case where $syscall \in \sum_T$ has been read and decomposed into 2 system object instances such as $\text{src}_x, \text{sink}_y \geq 1$ and dependency relationdep_z: $\text{src}_x \rightarrow \text{sink}_y$in which$\text{src}_x$ was shown as $x^{t\square}$ Object occurrence src εO_T and sink$y^{t\square}$ Object representation sink $\in O_T$. The timestamps of $syscall, dep_{c,}and\text{sink}_j$ and t_syscall, t_dep_z, t_src_x, and t_$sink_y$. The t_dep_z receives t_$syscall$from$syscall$. The indexes src_x and $sink_y$ are analyzed before the addition of V.

b. Supposed that$\forall src_x, sink_y \in$ V: Maximum instance indicessrc and $sink$, x_{max} and y_{max}, i, j≥ 1.

c. When$\exists src_z$V, z≥1,$t\square enx = x_{max}$, and t_src_x remainsidentical; else, x = 1, and t_src_x is changed to $t_{syscall}^{i} If i \ni sink_i$ V, i≥1, then$y = y_{max}$; else, y = 1. $ti_{sink_y}^{j}$has changed to $t_{syscall}$If j is equal to 2, then E is equal to E U{dep_s: $sink_y - 1 \rightarrow sink_y$}.

d. When a→b ∈ E and bc∈ E, Then, c is transitively linked to a new instance of a.

To begin with, the System Object Graph has the potential to show the appropriate causality relation in infection because it implies time data, which is connected to specific instances. Second, the instance graph may be divided into the cycles of SODG.

Algorithm 5.1 Generate Object Instance Graph

```
Input:
   D : system object dependency set
Output:
   G(V, E) : instance graph
1. Initialize empty vertex set V
2. Initialize empty edge set E
3. For each dependency dep: src → sink in D do:
    # Step 3a: Find most recent instances
    src_k ← most recent instance of src in V
    sink_z ← most recent instance of sink in V
    # Step 3b: Handle sink not in V
    If sink_z  V then
       sink_1 ← create new instance of sink
       V ← V ∪ {sink_1}

       If src_k  V then
          src_1 ← create new instance of src
          V ← V ∪ {src_1}
          E ← E ∪ {src_1 → sink_1}
       Else
          E ← E ∪ {src_k → sink_1}
       End if
    # Step 3c: Handle sink already in V
    Else
       sink_(z+1) ← create new instance of sink
       V ← V ∪ {sink_(z+1)}
       E ← E ∪ {sink_z → sink_(z+1)}
       If src_kV then
          src_1 ← create new instance of src
          V ← V ∪ {src_1}
          E ← E ∪ {src_1 → sink_(z+1)}
       Else
```

```
            E ← E ∪ {src_k → sink_(z+1)}
        End if
    End if
4. End for
5. Return G(V, E)
```

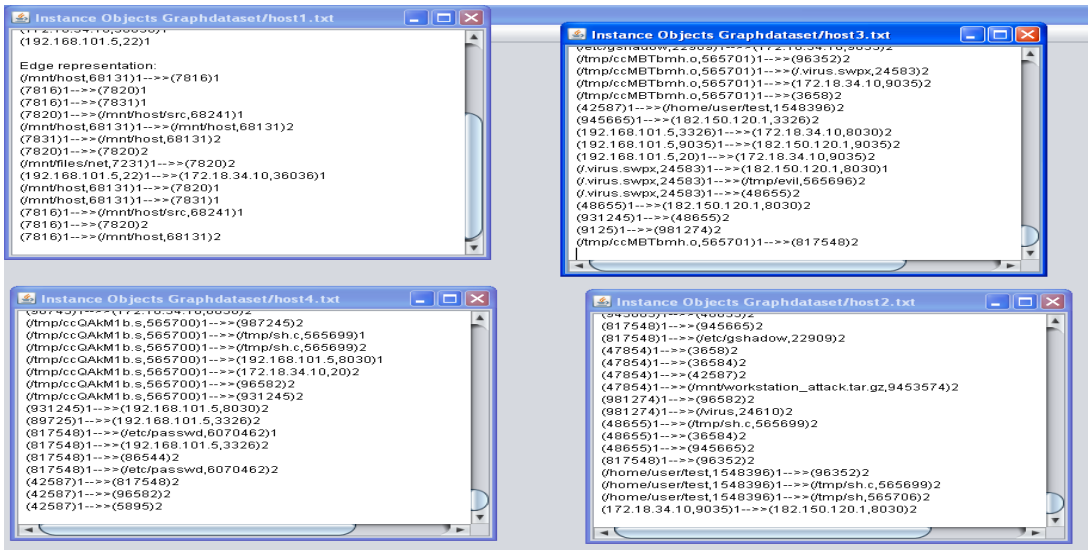

Figure 5.4 Object Instance Graph

vii. Instance Graph Pruning: Pruning of instance graph makes processing easier and faster. Redundant system object records and dependencies are removed. Even though this have been a different system call, it is not unusual that certain dependency similar may happen at other times between any two system objects.

viii. Incorporate Evidence: Within Infected nodes are established beforehand in the environment and assigned probabilities reflecting likelihood of being compromised, aiding in the identification of potential attack paths. The module embeds the evidence within an instance-based graph structured around a grounded BPNN. Each object instance is labeled as compromised, or a Local Observation Model (LOM) node is attached as a child to model observational uncertainty.

ix. Enhanced Back Propagation-Neural Network: Enhanced BPNN is an advancement of the classical BPNN that is a form of ANN that is popularly applied in pattern recognition and ML exercises. The Enhanced BPNN is specifically modified to improve the accuracy and effectiveness in terms of identifying the zero-day attack

paths. Other features like momentum and adaptive learning rate are added to this improvement. Momentum is important to ensure that the model does not get stuck in local minima by providing the optimization process with a sense of inertia so that it is more effective to wander across the solution space. Adaptive learning rate varies learning rate through the training, which guarantees the model converges more efficiently and quickly. Together with these improvements, the Enhanced BPNN will be more robust and will be able to find the complicated patterns in the data, which is pivotal in detecting subtle and dynamic features of zero-day attack trails.

The EBPNN learns the probability of the propagation of zero-day attacks based on the instance graph. It has both an input layer and several hidden layers, and the output layer. Momentum is adopted to avoid local minima in training. Adaptive Learning Rate: to approach the solution in a faster and more stable manner. This approach use training set in generating probability by the Neural Network. Based on this training, the affected rate of the test node is computed and illustrated in Figure 5.4.
It is inclusive of an input layer, one or many hidden and one output layer.

Set of Sample say

D= $\{(x_1, t_1), (x_{2,t_2}) \ldots (x_{i,t_i}) \ldots (x_{n,t_n})\}$ --------- (5.1)

Withx_{i,t_i} represents $i^{t\square}$input vectors being trained, and t_i represents similar output one.

The BP neural network makes it possible to write the predicting results as:

$\widehat{t_i} = \sum_{j-1}^{M} \omega_2(j)\tanh(\sum_{j-1}^{r} \omega_1(j,k)x_{jk} + \theta_1(j)) + \theta_2$ ----- (5.2)

P denotes the number of input neurons and M represents the number of hidden neurons in the model$x_j = [x_{jk,}k = 1,2,\ldots P]$. The weights are: $\omega_1(j,k)$ the weight of input to the hidden layer, $\omega_2(j)$ the weight of the hidden to the output layer. $\theta_1$and $\theta_2$are the bias term.

The network configuration includes the number of layers, the number of neurons per layer, each neuron's activation function, and the correlation (CV) between neural units. Weight adjustments are performed to minimize the error, bringing it as close to zero as possible.

$$E_p = \frac{1}{2}\sum_{req}^{n}(t_i\text{-}\widehat{t_i})^2 = \frac{1}{2}\sum_{err}^{n} e_i^2 \text{ ------ (5.3)}$$

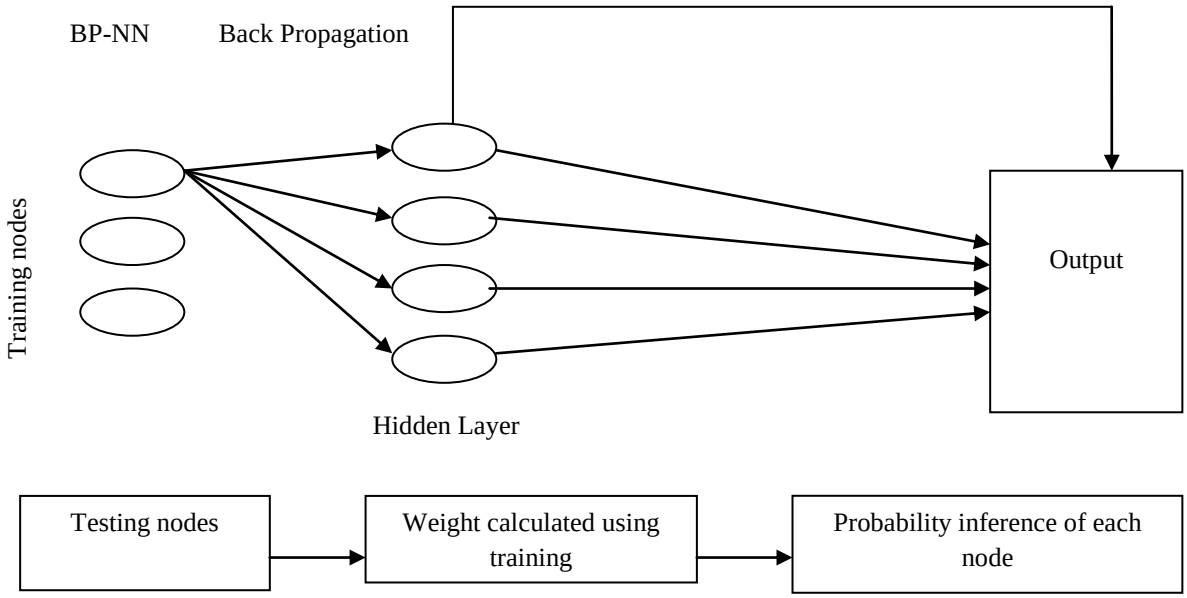

Figure 5.5 Back Propagation-Neural Networks

It serves to assess the BP Neural Network parameters, with weight decay included in the network error function as defined below.

$$\text{F (W)} = \beta\overline{E_p} + \alpha\overline{E_\omega} \text{ -------- (5.4)}$$

Where, α, β are hyperparameters, $E_\omega$is weight decay term. Where, α,β are hyper parameters, E_ω is the weight decay.

Using $e_\infty = \frac{1}{2}\sum_{j=1}^{m}\omega_{1,2}\,(f)^{2\prime}$as hyperparameters,Weights from the input to hidden layer and from hidden to output layer are indicated by Weight CV, while the Bayesian posterior probability function is expressed as CV.

$$\text{P }(\omega|D,\alpha,\beta,H) = \frac{p(D|\omega,\beta,H)p(|\omega|\alpha,H)}{p(D|\alpha,\beta,H)} \text{ -------- (5.5)}$$

Through weight adjustments, the hidden units adjust weights to capture the most relevant features of the task domain. A Backpropagation Neural Network (BPNN) is composed of three layers: 1) Input Layer, 2) Hidden Layer(s), and 3) Output Layer. The number of hidden layers and the number of neurons in each layer are determined by the complexity of the problem. The learning process of a BPNN proceeds in two main steps:

Step 1 Forward Propagation: Here, based on the inputs and the existing weights, the outputs are obtained. To do such calculation, the net excitation calculated by each hidden unit and each output unit is dependent on:

a. The values of values of the prior layer units which connect to the unit under consideration.

b. Masses of the thickness before layer and unit under consideration.

c. Minimum value on the unit in consideration.

The EBPNN applied in the discovery of the zero-day attack path and net excitation of each neuron of EBPNN is computed as the weighted sum of the inputs. This net excitation is also applied to an activation function which produces the neuron output. To allow learning based on the gradients in the process of learning, activation function must be continuous and differentiable. This step involves the activation of sigmoid at the hidden neurons in the EBPNN. The net excitation is scaled with the sigmoid to a range between 0 and 1 to ensure the outputs of the nodes can be interpreted in a probabilistic manner and the nodes on the path between attack paths can be easily described. It is defined as (5.6).

$$S_c(x) = \frac{1}{1+e^{-cx}} \text{ ----- (5.6)}$$

Step 2 Backward Propagation: In this step, it compares the actual output against the targeted output of each output unit to come up with an error. This is an error that is back propagated to the earlier layer which is the hidden layer. The error at each node in hidden layer NNN is calculated, and similarly, the errors for nodes in the previous hidden layer N−1N-1N−1 are determined. These calculated errors are subsequently used to update the weights, with the objective of minimizing the output error. The forward and backward passes are repeated iteratively until the error reaches an acceptable minimum.

x. Zero-Day Path Identification: The probabilities that are inferred to understand zero-day attack paths are used, and nodes with high probabilities and edges connecting the instance graph are identified. Zero-day attack detection systems involve the identification of zero-day paths, which are very important. It is an action of tracking and examining the steps or events followed by an attacker to use zero-day vulnerability and inflict damage on a system or a network. The security professionals may use these paths to understand the tactics, techniques, procedures (TTPs) used by the attacker to be more effective in mitigation and

response strategies. These paths are illustrated in Figure 4.5.It is likely to cause motivation on these nodes in instance graph both on its own and on its descendant. These highlighted nodes are in the chain of infection spread and thus it has to be kept and represented in Figure 4.6 (a) and (b). A high probability value, referred to as the tuning parameter (threshold), is used to control the lower probability bound. The resulting graphs are stored in GraphML format and can be visualized using the yEd editor.

Algorithm 5.2 Identify Zero-Day Attack Path

```
Input:
   G(V, E)  : instance graph
   v V    : starting vertex
Output:
   G_z(V_z, E_z) : zero-day attack path
Function DFS(G, v, direction)
   Mark v as visited
   find_high_probability ← False
   # Determine next vertex and flag based on direction
   If direction = ancestor then
      next_vertices ← parent nodes of v where next_v → v ∈ E
      flag ← high probability ancestor
   Else if direction = descendant then
      next_vertices ← child nodes of v where v → next_v ∈ E
      flag ← high probability descendant
   End if
   # Traverse all next vertices
   For each next_v in next_vertices do
      If next_v is not visited then
         If prob[next_v] ≥ threshold OR next_v matches flag then
            find_high_probability ← True
         Else
            DFS(G, next_v, direction)
         End if
      End if
   End for
   # Mark current vertex if high probability path found
   If find_high_probability = True then
      Mark v with flag
   End if
End Function
# Main Procedure
V_z
```

E_z
For each vertex v in V do
    DFS(G, v, ancestor)
    DFS(G, v, descendant)
End for
# Construct zero-day attack path vertices
For each vertex v in V do
    If prob[v] ≥ threshold OR
      (v marked as high probability ancestor AND v marked as high probability descendant) then
        V_z ← V_z ∪ {v}
    End if
End for
# Construct zero-day attack path edges
For each edge e: v → ω in E do
    If v V_z AND ω V_z then
        E_z ← E_z ∪ {e}
    End if
End for
Return G_z(V_z, E_z)

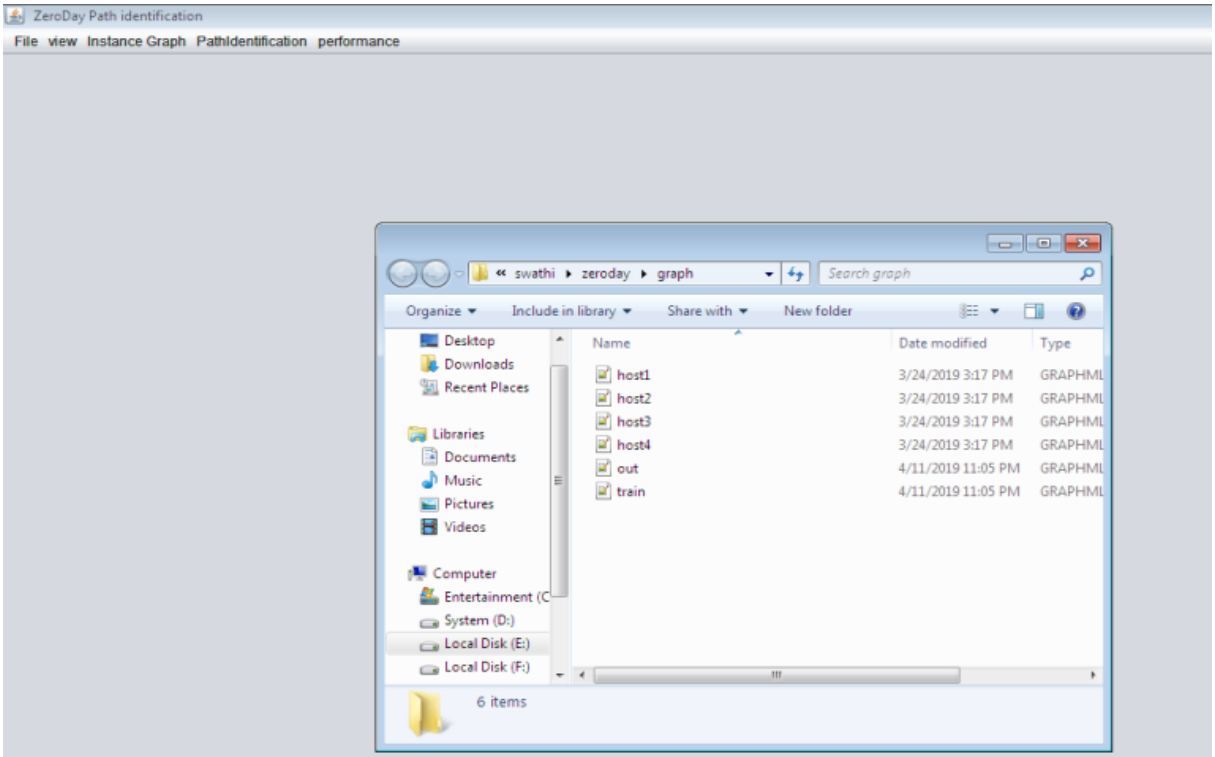

Figure 5.6 Zero-Day Attack Path Identification

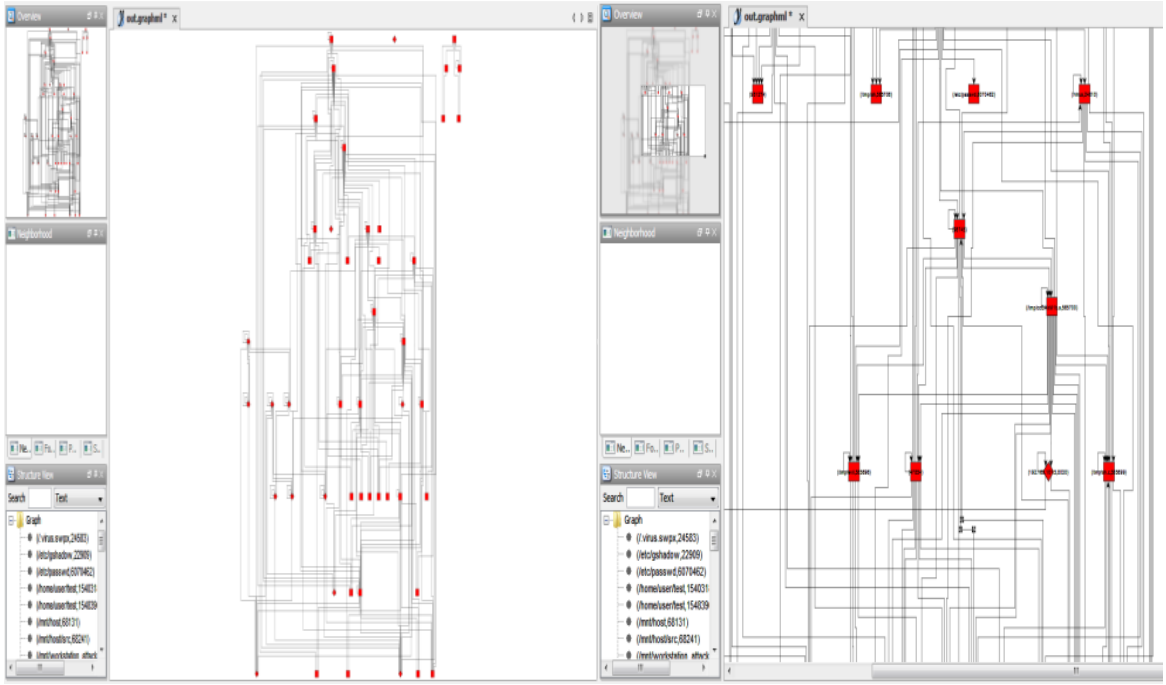

Figure 5.7 (a) and (b) is Yed Editor for Showing Output

5.5.2 High Probability Tuning Parameter (Threshold)

The threshold (tuning parameter) is the smallest probability value that is used to determine the probability that a node or an edge within the instance graph is a member of a zero-day attack path. The nodes/edges with probability values exceeding its threshold are selected and recognized as potential attack components only. Rationale: The threshold can be used to remove a large number of low-probability nodes of the model thereby removing much noise and possible false positives. In a probability threshold, the model eliminates doubtful or weak links and only shows those nodes that impact heavily or substantially to the spread of the infection.

Thus, identification of paths is theoretical, more efficient and quicker. Value Added: In the EBPNN model the best threshold value had been empirically determined by testing and validating it on simulated data using CloudSim, and alternative threshold values (0.6-0.9) were then tested to identify the threshold parameter (0.8, 80%) that gave the best balance on the recall (defending most attack nodes) and low false positives. With the accuracy of 99.0, the precision of 99.21, the recall of 99.34 as well as the F1 -score of 99.54 the EBPNN gave the best performance at this threshold level.

Table 5.1 Effect of Threshold on EBPNN Performance

Threshold (Probability)	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
0.6	97.5	96.8	97.1	96.9
0.7	98.2	97.5	98.0	97.7
0.8 (Optimal)	99.0	99.2	99.3	99.5
0.9	98.4	98.5	98.1	98.3

5.6 Experimental Setup and Results

This part discusses the setups and findings of the experiment.

5.6.1 Simulated Experiment

Cloud Sim is a simulation framework meant to run a simulation and modeling of cloud computing infrastructure and services. As a realistic testing tool to test the performance of ML models in the context of zero-day attack path identification, Cloud Sim can be utilized. It enables researchers and practitioners to model a number of features of cloud environments, including network traffic, resource distribution, and communication between different actors, such as attackers and victim systems. Such realistic simulation plays an important role in the testing and training of ML models in situations that are as close as possible to real-life cyber threat. With Cloud Sim together with the Enhanced BPNN, the research can enjoy the added advantage of a better depiction of the dynamism and changeability of the real world cyber threat, which in the end will enhance the a detection model for zero-day attack paths.

The given approach is realized with the help of a cloud simulator, in which case the programming language is Java and the database is SQL server. Java based IDEs like Net Beans and Eclipse are used together with CloudSim. It is possible to access the CloudSim library. The virtual machines known as Virtual Machine Monitor (VMM) are generated and executed by a software component. The estimation of the existing and proposed techniques is done through simulation parameters in this tool. It is a flexible tool that provides a generalized simulation framework for researching Cloud computing environments and application services. yEd is a widely used diagramming platform that offers a multi-document interface. Developed in Java, it is cross-platform and supports the creation of a wide range of diagrams, including entity-relationship diagrams, network diagrams, and

flowcharts. Users can also incorporate customized vector and raster graphics into diagrams. Various parameters are used to evaluate the performance of the proposed method, and the results are presented and analyzed in the following section.

5.6.2 Performance Metrics

To estimate the performance of the proposed work, the following metrics have been used and this has been depicted in Figure 5.7.

a) Accuracy- Accuracy presents the necessary related outputs that are utilized in classification

$$Accuracy = \frac{TruePositive+TrueNegative}{TruePositive+FalsePositive+TrueNegative+FalseNegative} \text{ --- (5.7)}$$

Accuracy is applicable to gauge general model accuracy. In zero-day attack detection, however, accuracy in itself can be deceptive, particularly with an imbalanced dataset (i.e. much more normal data than attack data). This is why it has to be supplemented by other measures.

b) Correctness- Correctness is described as a number of correct nodes by a total number of nodes.

Correctness = Number of Correctly Classified Nodes/Total Nodes ------ (5.8)

Correctness is that the prediction of the model is correct or factual - i.e. how close the outcome predicted is to the actual class or behavior of the input of the input, in a security-critical decision.

Misclassification- This represents the sum of false positives and false negatives relative to the total number of nodes.

c) Misclassification Rate = (False Positive +False Negative)/Total ----- (5.9)

Misclassification is the label on the model that is inaccurate i.e. when: An attack is called normal (False Negatives), or, Normal behavior is called an attack (False Positions).

Accuracy in detection is not sufficient in cybersecurity. Even one false negative (FN) (i.e. missed attack) can cause tremendous damage in a zero-day detection.

- Know the failures of the model and where it occurs.
- Enhance the robustness of the model by detecting data trends of errors.
- Optimize guides (such as OLFFOA in Phase 4) in order to reduce FN and FP.

d) $Precision_i = \frac{TP_S}{TP_S + FP_S}$ ---- (5.10)

Precision means that the model will not give false alarms (false positives). It is especially relevant to real-time systems (Phases 2 & 3) because incorrectly marking innocent actions as an attack may cause resource wastage or denial of service.

The recall for category (s) is defined as the proportion of correctly predicted samples of category (s) to the total number of samples of category (s) in the dataset, as expressed in Equation (5.11).

e) $Recall_i = \frac{TP_S}{TP_S + FN_S}$------- (5.11)

F-measurement can be calculated using the formula below.

Recall is used to measure the capability of the model to identify real attacks, though it can be infrequent. In the zero-day, a false positive is not as harmful as a false negative (miss). Therefore, recall is very essential in order to reduce threats that are not detected.

f) $F - Measure = 2.\frac{Precison.recall}{Nprecison+recall}$------- (5.12)

F1-score gives a fair comparison between the recall and precision. It is the most confident measure in zero-day detection when imbalanced data are concerned, since it does not exhibit the bias of accuracy and offers a more accurate idea of how a model can operate on the instances of the rare classes.

Table 5.2 Performance Metrics Comparison Table

Methods	Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Correctness (%)	Misclassification (%)
Existing	Bayesian	96.11	96.67	97.14	97.27	96.20	3.89
	Scalable Bayesian	97.12	97.27	97.47	98.38	97.30	2.88
	Probabilistic Bayesian-Net	98.20	98.22	98.21	98.95	98.25	1.80
Proposed	**EBPNN**	**99.00**	**99.21**	**99.34**	**99.54**	**99.10**	**1.00**

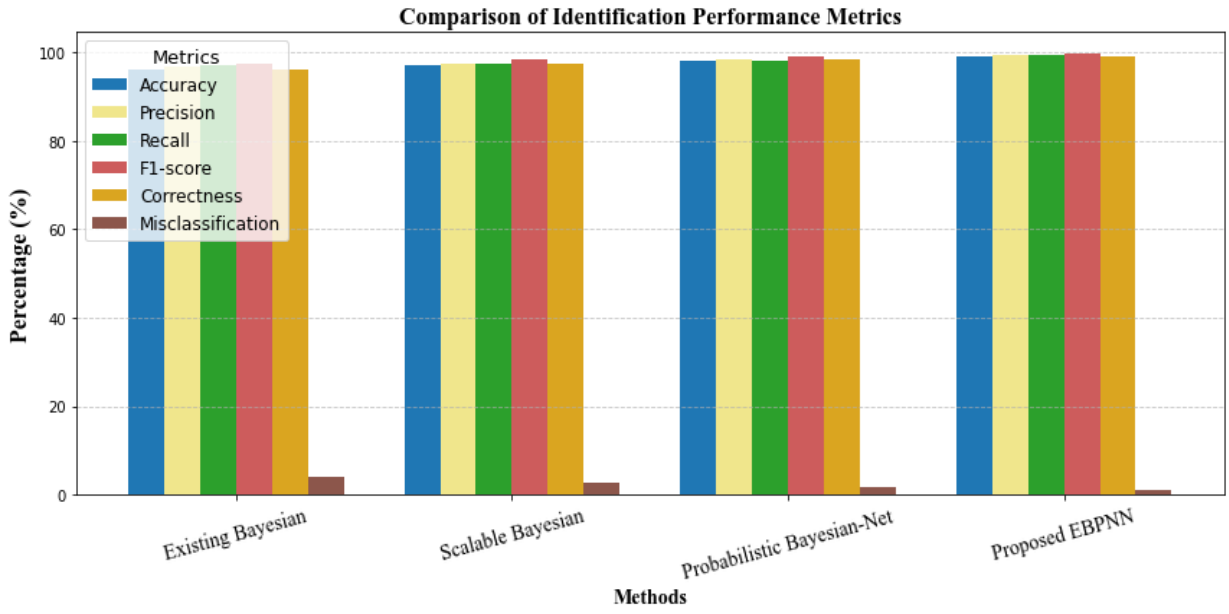

Figure 5.8 Performance Metrics Comparison Chart

The comparison of performance between some important measures to the EBPNN model and the existing Bayesian based algorithms is summarized in the table 5.2 and figure 5.8. The highest accuracy, precision, recall, F1-score, and correctness rate are the best performance of the current methods, Probabilistic Bayesian-Net, which decreases the misclassification. Nevertheless, the proposed approach based on EBPNN has a better result in all these metrics and achieves the 99% accuracy and 1% misclassification, which is significantly higher considering the detection potential. It implies that EBPNN addresses the drawbacks of the traditional Bayesian methods with greater complexity and lower error rates, and this gives credence to EBPNN as a prospective state-of-the-art model of cyberattack detection.

5.6.3 Percentage Improvement of Proposed Work

The percentage improvement quantifies how much the proposed method outperforms existing methods across Various evaluation metrics, such as accuracy, precision, recall, and F-measure, are considered, and the percentage improvement is computed using the following formula

$$Improvement\ (\%) = \frac{BPNN\ score - Baseline\ score}{Baseline\ score} \times 100 \text{ -------- (5.13)}$$

This metric can be used to assess the performance improvement of the proposed algorithm relative to existing algorithms.

Table 5.3 Comparison of EBPNN Performance Improvements

Metric	Bayesian → EBPNN	Scalable Bayesian → EBPNN	Probabilistic Bayesian-Net → EBPNN
Accuracy ↑ (%)	3.01	1.94	0.82
Precision ↑ (%)	2.63	1.99	1.01
Recall ↑ (%)	2.27	1.92	1.15
F1-score ↑ (%)	2.34	1.18	0.60
Correctness ↑ (%)	2.90	1.80	0.85
Misclassification ↓ (%)	2.89	1.88	0.80

There is a performance enhancement of proposed EBPNN over each existing methods, which is indicated by this table 5.2. The real existing and suggested comparisons are presented. The presented EBPNN in table 5.3 can be determined as being far more effective than existing Bayesian models. It offers a maximum error of 3.01 percent and 2.63 percent reduction in error indicating an increase in detection power. Compared with the advanced models, e.g. Probabilistic Bayesian Networks, EBPNN continues to register improvement in all performance indices, thus demonstrating itself to be powerful and effective in the monitoring of zero-day attack paths.

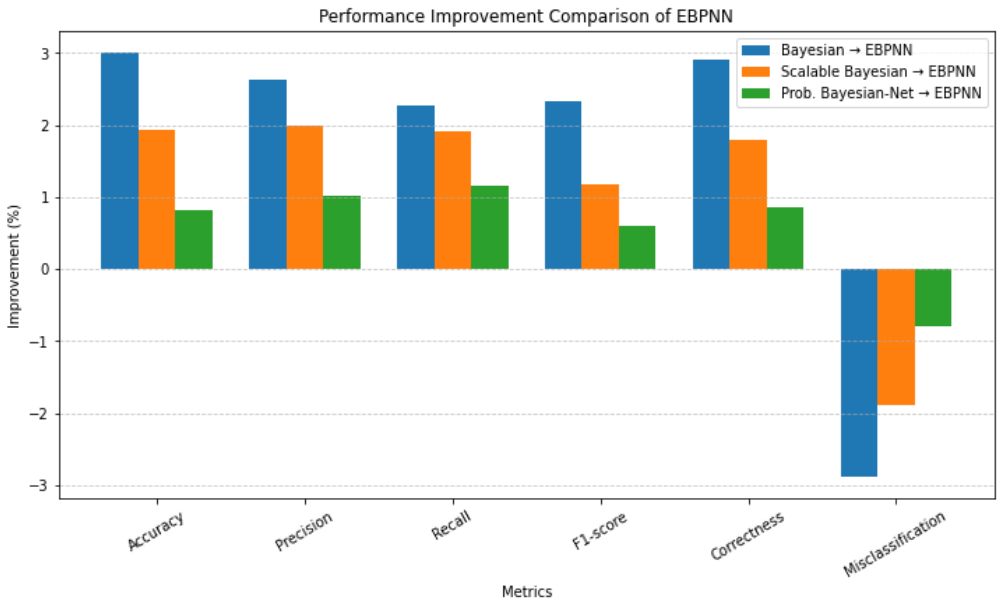

Figure 5.9 Performance Improvement Comparison Chart

As shown in Figure 5.10, the proposed BPNN model has been found to significantly improve over the existing models Sun et al. (2016) and EDCNN in four evaluation metrics (Accuracy, Precision, Recall, and F-measure) of the model. Evaluation metrics appear in the x-axis in this chart. The values of improvement are depicted on the y-axis.

5.7 Chapter Summary

This chapter introduced EBPNN model to find zero-day attack paths. This outlined the model structure, learning process and the reason why it works much better than the other methodologies based on Bayesian. The addition is the incorporation of adaptive learning rate and momentum in combination with the classic BPNN to achieve the enhanced convergence and pattern recognition. EBPNN was the best improvement of up to 3.01% of better accuracy and showed general improvement in all the measurements of evaluation compared to the existing approaches. The next chapter presents a zero-day vulnerability predictive model that is informed with the combination of Hybrid Game Theory and Neural Networks and attacker-defender dynamics. The model is a combination of EBPNN and active anticipation of threats prior to the spread.

Publications

- Swathy Akshaya M and Padmavathi G. Zero-day attack path identification using probabilistic and graph approach based back propagation neural network in cloud. Mathematical Statistician and Engineering Applications. 2022; 71.3s2, 1091-1106.