

Classification of malware using multinomial linked latent modular double q learning

Sheelavathy Veerabhadrapa Kudrekar, Udaya Rani Vinayaka Murthy

Department of School of Computer Science and Engineering, REVA University, Bengaluru, India

Article Info

Article history:

Received Jun 9, 2022

Revised Jul 29, 2022

Accepted Aug 4, 2022

Keywords:

Double q learning

Linked latent dirichlet

Malware attack detection

Multinomial

Network behavior

ABSTRACT

In recent times, malware has progressed by utilizing distinct advanced machine learning techniques for detection. However, the model becomes complicated and the singular value decomposition and depth-based malware detectors failed to detect the malware significantly with minimum time and overhead. This paper proposes a multinomial linked latent dirichlet and modular double q learning (MLLD-MDQL) to efficiently detect malware based on the network behavior patterns. First, multinomial linked latent dirichlet network behavior extraction (ML-LDNBE) is applied to the input network for anomaly detection that extracts the behavior based on the network pattern. The behavior is extracted which are grouped to perform on the path protocol for analyzing repeated behaviors. Finally, the modular double q learning malware classification model is the grouped behaviors for significant malware detection. The effectiveness of proposed MLLD-MDQL method is compared with existing models. The results obtained by the proposed method show that the model combined with the machine learning (ML) significantly determined malware detection time and also reduced the false positive rate (FPR). The results showed that the false positive rate is significantly reduced by 42% for the proposed method better when compared to the existing behavior based malware detection model that obtained 62% of FPR.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Sheelavathy Veerabhadrapa Kudrekar

Department of Computer Science and Engineering, REVA University

Rukmini Knowledge Park, Kattigenahalli, SH 104, Srinivasa Nagar, Bengaluru, Karnataka 560064, India

Email: sheela.kv@reva.edu.in

1. INTRODUCTION

In computer security, malware detection is prospering due to its devastating intents. In recent years, it has become an immense ultimatum to make the system secure from malware attacks [1]. The rapid growth of the malware complication was having an issue for both computer security and the network [2]. In the existing research Khraisat *et al.* [3], three prominent features were processed to develop a malware classifier model [4]. Initially, strings in printable formats were processed through mining results the high dimensional features which are on the string. Next, to reduce the dimension involved, singular value decomposition was applied [5]-[7]. Shannon entropy was evaluated to consider both the randomness of the application programming interface (API) and printable string information (PSI) features [8], [9]. Moreover, behavioral features such as key modification, file operations regarding the registry and network activities for malware detection [9]-[11]. The features are combined and summed up for training the feature set emerged with the machine learning-based algorithms for malware classification [11]-[13]. With this, the malware detection accuracy was recorded to be high [14].

A malware based on behavior chains was proposed in [15]. The proposed method monitored the behavior points based on calls made by API. Then the model utilized the calling progression of the behavior points performed at runtime to assemble a behavior chain [16]-[18]. Finally, a depth detection model has utilized based on long short term memory that in turn differentiated between malicious behaviors that were observed from the behavior chains [19]-[22]. With this, the malware detection accuracy was said to be improved with a minimum false positive rate [22]-[24].

This paper introduces a method to detect malware attacks. In this paper, a behavior-based machine learning technique called multinomial linked latent dirichlet and modular double q learning (MLLD-MDQL) is designed [25]-[29]. The model takes advantage of network behavior pattern extraction and behavior classification. The existing researchers used ML models for malware classification and analyzed the problems arose in them. Singh and Singh [30] detected the malware software to analyze based on the artifacts using ML algorithms. The malware detection was performed based on the behavior of the research. The run time features were extracted by analyzing the dynamic environment based on the Cuckoo search algorithm. However, the developed model failed to achieve better efficiency concerning the context of the processed features.

Similarly, Zhang *et al.* [31] developed a depth detection model for malware detection based on the behavior chains. The developed model was related to the behavioral points and has proposed a depth detection method with respect to the behavior chains. The developed model monitored the behavior based on the API calls and designed those points in the calling sequence at the run time for behavior chain construction. The developed model detected the malicious behavior based on the long short term memory (LSTM) that required to analyse the model based on the behavior chain. Additionally, Jeon *et al.* [32] developed a convolution neural network (CNN) for dynamic analysis and detection of IoT malware. The developed DAIMD scheme learned the IoT malware using the CNN-based model for analyzing the cloud environment. Thus, the implementation of the model was important to detect the IoT malware based on the hybrid analysis technique that utilized both dynamic and static techniques further.

Similarly, Zhao *et al.* [33] developed a hybrid Gram model for feature extraction to detect malicious behavior using machine learning (ML). The developed model was constructed based on the dynamic feature for malware analysis as it used a novel feature extraction method for the H-gram to evaluate the cross-entropy. It provides the continuous overlapping of subsequences that implements the semantic segmentation for the API calls and instructions. However, the extracted feature sequence was not enough that show a lower degree of discrimination. Also, Liu *et al.* [34] developed a machine learning algorithm for automated malware classification and the decision making utilizes the features for the classification. The developed model identified the suspicious malware and classified the new malware. The detection modules used shared nearest neighbor (SNN) based model for clustering and to discover the families of the new malware.

The results showed that the developed model improved the accuracy for the effective classification but failed to discover the new malware effective and thus, Pattee *et al.* [35] developed an intelligent malware detection and design alternatives for performance monitoring counter. The developed model improved the importance of selecting the features for malware detection that showed statistical differences among the malware workloads and benign workloads were characterized on the basis of information from counters performances. However, the detailed architectural design for the dedicated accelerator was required for providing efficiencies with the chip area power, and processing time. Also, Botacin *et al.* [36] developed a signature based malware detection system for accelerating the hardware-enhanced antivirus engine (HEAVEN). The workflow consisted of a hardware-assisted signature that matched the process of first step (triage) that is fast, and invokes with the software-based AV. At the time of software suspiciousness an unknown hardware signature is generated for malignity. The mechanisms were claimed with flexibility than HEAVEN as it was not limited to signature matching. Yet, the model was not often discussed when it comes for flexibility showed improvement in terms of performance cost that was not required as the signatures were not enough.

The multinomial linked latent dirichlet (MLLD) to extract network behavior and then classify by applying modular double q learning (MDQL). In short, the main contributions are as follows:

- To construct a novel network behavior pattern model known as MLLD network behavior extraction model that combined linked latent dirichlet allocation with behavior graphs of API for malware detection.
- The MLLD-MDQL was developed to analyse the modular double q learning classification model which automatically acquires high-level representations of malware behaviors.
- To obtain experimental results exhibiting the proposed method extracts more relevant conceptual features and helps to reduce the time, overhead and false-positive involved in malware detection.

The organization of the research paper is given as follows: Section 2 explained the proposed method with malware behavior. Section 3 shows the results for the proposed method which analyzes the malware behavior for the simulation of the results. The conclusion of this research is given in Section 4.

2. RESEARCH METHOD

The behavior-based malware detector on the other hand distinguishes network behaviors with the aid of monitoring tools and decides whether the program is malware. The MLLD-MDQL is split into two stages which are observed and obtain the dimensionality reduced network pattern behaviors. Next, malware attack detection is made by means of a robust classification model. A detailed description of the proposed method is followed with the below dataset description.

2.1. Dataset description

In this work, the malware detection based on the network behavior patterns is made by applying <https://www.kaggle.com/anushonkar/network-anomaly-detection-dataset>. The dataset includes basic features, $BF = \{bf_1, bf_2, \dots, bf_9\}$, content features $CF = \{cf_1, cf_2, \dots, cf_{13}\}$ and time-related traffic features $TTF = \{ttf_1, ttf_2, \dots, ttf_{19}\}$ respectively.

2.2. Multinomial linked latent dirichlet network behavior extraction

Network behavior cites the pursuits of both the network and the users' control to work on it. To accurately measure its network's security, network behavior patterns have to be analyzed and observe for any malware that stipulates a security threat. This malware detection through network behavior patterns not only assists in alleviating security issues but also inspects present-day and prevailing behavior to obtain a comprehensive sketch of network security.

In this section, a malware detection using behavior association is presented. The network behavior patterns are extracted with the aid of the MLLD model. Figure 1 shows the block diagram of the multinomial linked latent dirichlet network behavior extraction (ML-LDNBE) model.

As illustrated in Figure 1, the input is acquired from <https://www.kaggle.com/anushonkar/network-anomaly-detection-dataset>. The main objective of the above model remains constant in extracting the behavior based on the network patterns of malicious code based on the API monitoring model fuses the behavior. Next, with data dependence, an association between behaviors is confirmed.

Finally, the MLLD allocation model is utilized in extracting semantically and globally related behaviors from sample programs that permit the set of observations. This is obtained by hidden groups that explain why certain behaviors of the network traffic data are similar. The linked type of allocation is performed by using multinomial latent dirichlet for behavioral network-based feature extraction. The faster access time with less memory overhead better the ensure.

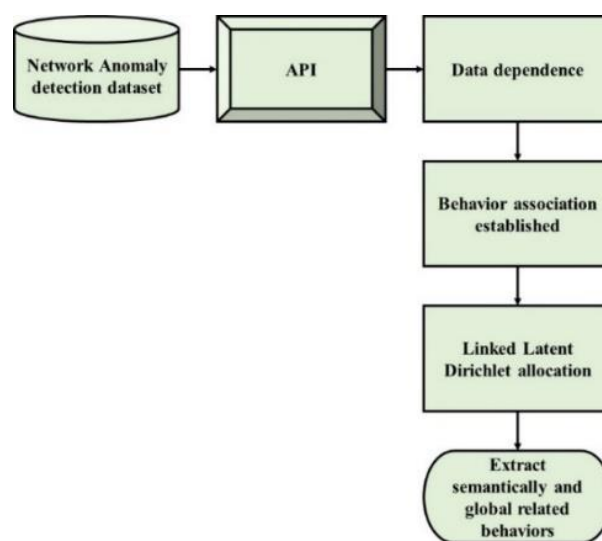


Figure 1. Block diagram of multinomial linked latent dirichlet network behavior model

Let us consider a digraph ' $G(V, E)$ ' to denote the behavior association. Here, the node ' V_i ' denote the behavior and a directed edge ' (V_i, V_j) ' denotes the data dependence relationship, where ' $V_i \rightarrow V_j$ ' denotes ' V_j ' that depends on ' V_i '. Network behavior refers to multiple approximately close behavior patterns, like, the number of file creation, source bytes, duration and so on. To ease the interpretation of the upcoming x algorithm, let us utilize a 4-tuple behavioral pattern model, which is mathematically expressed in (1).

$$A = \{BS, IB, OB, CB\} \quad (1)$$

From (1), ' BS ' denotes the behavior states (i.e., success, failure, or idle state), ' IB ' denotes the input behavior, ' OB ' denotes the output behavior and ' CB ' represents the combination behavior respectively. The behavior state is an API where the program is called to accomplish a task as shown in Figure 1, the ML-LDNBE in the present work is used to extract the dimensionality reduced network behavior features (i.e. patterns). The ML-LDNBE model assumes that each time-related feature represents the probabilistic distribution through the latent distribution. The content is performed all the time which is related to features shared before common Dirichlet.

Each of the latent content is related to the ML-LDNBE features which are represented with the probabilistic distribution over a basic level of features which are having basic distributions for sharing the content in common to a prior. Thus, the content of the network vectors was linked statistically for linking the structure among basic and content features. The time-related traffic and content features are therefore summarizing the dimensionality-reduced network behaviors.

Given a set of network connection vector ' $V \in BF, CF, TTF$ ' consisting of ' n ' traffic features, with network connection vector ' n ' having ' V_n ' traffic features, ML-LDNBE models ' V ' according to the subsequent generative procedure. Let us select a multinomial distribution ' ϕ_{cf} ' for content features ' $cf (cf \in \{1, 2, \dots, CF\})$ ' from a Dirichlet Distribution with criterion ' α ', multinomial distribution ' ϕ_{tff} ' for time features ' $tff (tff \in \{1, 2, \dots, TTF\})$ ' from a Dirichlet Distribution with criterion ' β ' and finally, a multinomial distribution ' ϕ_{bf} ' for basic features ' $bf (bf \in \{1, 2, \dots, BF\})$ '. Moreover, content feature ' δ_i ' from ' ϕ_{cf} ' and a basic feature ' δ_j ' from ' ϕ_{bf} '.

In the above generative process, basic features in a network connection vectors are observed with the other variables that are latently represented as ' ϕ ' and hyperparameters ' α ' and ' β ' respectively. To obtain the dimensionality reduced network behavior feature pattern, an exchangeability hypothesis is utilized. The hypothesis assumes that the integrated distribution between time and content related features, content and basic network features is uniform to permutation. Here, to handle the dimensionality reduced network behavior feature pattern, the API function is used and the behavior is said to be handled.

Let us assume that ' P ' is a permutation of the network features from ' 1 ' to ' N ', then, a finite set of random network features ' $\{cf_1, cf_2, \dots, cf_m\}$ ' are also said to be interchangeable. This is mathematically expressed as given in (2).

$$P(cf_1, cf_2, \dots, cf_m) = P(cf_{\pi(1)}, cf_{\pi(2)}, \dots, cf_{\pi(m)}) \quad (2)$$

From (2), ' $\pi(\cdot)$ ' refers to the permutation function on the network features ' $\{1, 2, \dots, m\}$ '. Then, the probability of observed network data ' OD ' is evaluated as given in (3).

$$Prob(OD|\alpha, \beta) = \prod_{i=1}^M Prob(\theta_{TTF}|\alpha) \left[\prod_{i=1}^M \frac{Prob(\delta_i|\theta_{TTF})}{Prob(\delta_j|\delta_i, \beta)i\delta_i} \right] \quad (3)$$

From (3), the generative process correlates to the cooperative distribution of the latent content ' $[\prod_{i=1}^M Prob(\delta_i|\theta_{TTF})Prob(\delta_j|\delta_i, \beta)i\delta_i]$ ' and observed variables ' $\prod_{i=1}^M Prob(\theta_{TTF}|\alpha)$ ' for time related features. The pseudo code representation of Multinomial Linked Latent Dirichlet Network Behavior is given as Algorithm 1.

As given in the above Multinomial Linked Latent Dirichlet Network Behavior extraction for malware detection, network behavior patterns are analyzed using content related, basic and time-related traffic features. With these features as input, first, a 4-tuple behavior pattern is formulated. Followed by which, permutation of network features is performed for a finite set of random network features. Finally, with linked data dependence, dimensionality reduced network behavior features (i.e. patterns) are extracted with minimum time and overhead.

Algorithm 1 multinomial linked latent dirichlet network behavior

Input: basic features ' $BF = \{bf_1, bf_2, \dots, bf_9\}$ ', content features ' $CF = \{cf_1, cf_2, \dots, cf_{13}\}$ ', time-related traffic features ' $TTF = \{ttf_1, ttf_2, \dots, ttf_{19}\}$ '
Output: Network behavior features ' B '
 Step 1: **Initialize** content feature multinomial distribution ' ϕ_{cf} ', time feature multinomial distribution ' ϕ_{ttf} ', basic feature multinomial distribution ' ϕ_{bf} ', criterion ' β ', ' α '
 Step 2: **Begin**
 Step 3: **For** each basic feature ' BF ', content features ' CF ' and time-related traffic features ' TTF '
 Step 4: Formulate 4-tuple behavioral pattern model using Eq. (1)
 Step 5: Evaluate permutation of the network features using Eq. (2)
 Step 6: Obtain probability of observed network data ' OD ' using Eq. (3)
 Step 7: Return dimensionality reduced network behavior features (B)
 Step 8: **End for**
 Step 9: **End**

2.3. Modular double q learning malware classification model

With the extracted network behavior patterns, malware attack detection is made in this work by applying modular double q learning malware classification model. Here, the grouping is performed which is followed with the classification approach. Grouping is performed based on path and protocol type for learning repeated behaviors. By considering what action to be taken under what circumstances, it does not require a process of the environment, without requiring adaptations. With double q learning, the classification is performed for different policies, where the value referring to the behavior is evaluated reduces false-positive rate.

As the number of agents (i.e. connection in network connection vector) increases, the exponential values also increase in the overall dimensions (i.e. number of connections), increasing the memory drastically. To address this issue, the modular function is utilized that splits a larger number of connections into sub-problem. In the action selection stage, each learning module provides Q-values for corresponding actions. The behaviors ' B ' and the paths (i.e. service '*Service*' from basic features), protocol (i.e. protocol type '*Protocoltype*' from basic features) are identified. The score is designated based on the protocol ('*Protocoltype*') and path ('*Service*'). For instance if ' $a_i \in V_i$ (i.e., network connection vector)' consists of ' B_1, B_2, \dots, B_n ' behaviors.

$$a_i(SP) = \frac{B_1(SP)+B_2(SP)+\dots+B_n(SP)}{n} \tag{4}$$

$$a_i(SS) = \frac{B_1(SS)+B_2(SS)+\dots+B_n(SS)}{n} \tag{5}$$

From (4) and (5), ' SP ', ' SS ' refers to the score for protocol type and score for service respectively with ' B_1, B_2, \dots, B_n ' representing the behaviors. Scores are calculated for each behavior. The score is categorized from '1' to '5', where '1' refers to the associated behavior which is normal, and '5' refers to that the associated behavior which is risky and probably is malware.

With the behavior grouped results, next, Double Q Learning is applied to the measured score for respective service ' $a_i(SS)$ ' and protocol ' $a_i(SP)$ ' to return different classification results. Therefore, four different types of malware classifications are made efficiently. From the name itself, Double Q Learning two estimators (i.e., service ' $a_i(SS)$ ' and protocol ' $a_i(SP)$ ') are utilized instead of one estimator for each state-action pair where the two values ' QP ' and ' QS ' are used.

Here, the action ' a ' is picked and based on ' $QP(s, \cdot)$ ' and ' $QS(s, \cdot)$ ', the final reward ' r ' and state ' s' ' are obtained. To start with either ' P ' or ' S ', the values are updated. If an update is made on ' P ', then corresponding action is represented as shown in (6).

$$a' = ARGMAX QP (s', a) \tag{6}$$

By finding the action ' a' ' from the above Eq. (6), next ' QP ' is updated and is mathematically formulated as given in (7).

$$QP(s, a) = QP(s, a) + \alpha [R + \gamma QS(s', a') - QP(s, a)] \tag{7}$$

Similarly, if an update is made on ' S ', the corresponding action is represented as shown in (8).

$$b' = ARGMAX QS (s', a) \tag{8}$$

By finding the action ‘ b' ’ from the above Eq. (8), next ‘ QS ’ is updated and is mathematically formulated as given in (9).

$$QS(s, a) = QS(s, a) + \alpha[R + \gamma QP(s', a') - QS(s, a)] \quad (9)$$

With the above actions evolved, four different types of malware are detected. As given in the above modular double q learning algorithm, two processes are performed. They are behavior grouping and malware classification. First, behavior grouping is done via modular function based on two estimators, protocol and service. Next, malware attack detection is made using double q learning separately for protocol and service. Condition checking with the respective threshold is validated to observe malware attack detection in the early stage. Therefore, by applying different policies for each update, accurate detection is said to be made with a minimum false positive rate.

3. RESULTS AND DISCUSSION

In this section, experimental analysis of the multinomial linked latent dirichlet and modular double q learning (MLLD-MDQL) method for malware attack detection is presented. In this section, the performance of the proposed MLLD-MDQL is compared with the state-of-the-art methods, Behavior-based malware detection [30] and Malware based on behavior chains [31] using network anomaly detection dataset and implemented in Java. Table 1 results attack detection time observed for three different methods, MLLD-MDQL, behavior-based malware detection [30] and malware based on behavior chains [31] and Figure 2 shows attack detection time graph with different connections.

Table 1. Attack detection time results using MLLD-MDQL, behavior-based malware detection and malware based on behavior chains

Count	Attack detection time (ms)		
	MLLD-MDQL	Behavior-based malware detection	Malware based on behavior chains
15	6.825	9.375	11.025
30	8.15	10.435	13.455
45	9.355	11.315	16.135
60	11.215	15.435	19.235
75	13.345	17.135	21.355
90	15.255	19.215	24.55
105	18.135	21.435	28.155
120	21.435	25.555	30.355
135	23.455	28.155	33.455
150	25.525	30.235	35.155

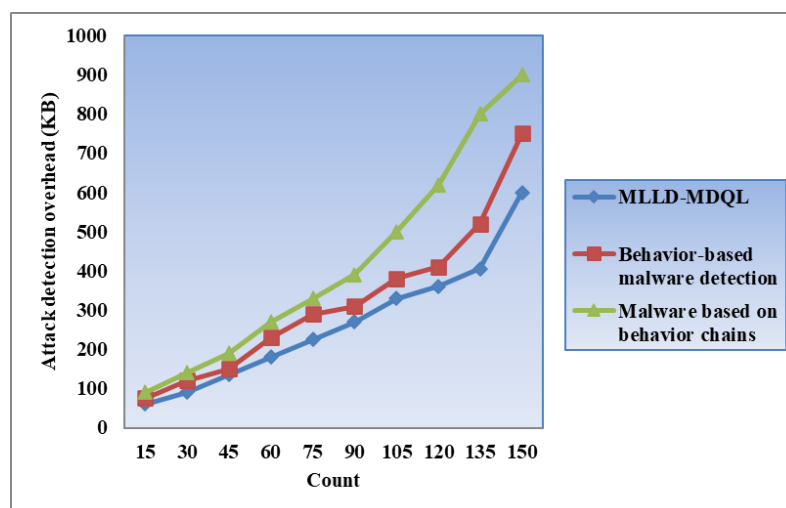


Figure 2. Attack detection overhead graphs with different connections

3.1. Quantitative analysis

3.1.1. Case 1: performance measure of attack detection time

The behavior based malware attack detection with respect to detection time is evaluated at significantly. The attack detection is referred to as the time consumed in malware detection based on the network behavior aspect. Mathematically, the equation is formulated as shown in (10).

$$ADT = \sum_{i=1}^n Count_i \times Time [AD] \quad (10)$$

From (10), the malware attack detection time based on the network behavior aspect ' $Time [AD]$ ' is measured based on the number of connections established as ' $Count_i$ ' and are measured in terms of milliseconds. Table 1 results attack detection time observed for three different methods, MLLD-MDQL, behavior-based malware detection [30] and malware based on behavior chains [31] and Figure 2 shows attack detection time graph with different connections.

3.1.2. Case 2: performance measure of attack detection overhead

The second metric analysed in this research work is the attack detection overhead. The attack detection overhead represents the overhead incurred while detecting malware attacks. The mathematical expression for the attack detection is expressed as shown in (11):

$$ADO = \sum_{i=1}^n Count_i \times Mem [AD] \quad (11)$$

From (11), the attack detection overhead ' ADO ' is obtained through many connections used to simulate purpose ' $Count_i$ ' and the memory consumed for detecting attacks ' $Mem [[AD]]$ ' during attack detection. The term kilo bytes (KB) is used to evaluate the model. The results obtained for the attack detection observed overhead for 3 distinct methods as MLLD-MDQL, behavior-based malware detection [30] and malware based on behavior chains [31].

In Figure 2, performances of the attack detection overhead are shown using line graphs for three different methods such as MLLD-MDQL, behavior-based malware detection [30] and malware based behavior chains [31]. The reason behind the improvement is the application of the Multinomial Linked Latent Dirichlet Network Behavior extraction algorithm. By applying this algorithm, network behavior patterns are analysed through content, basic and time-related traffic features. By considering these features, a 4-tuple behavior pattern is organized and permutation is performed for a finite set of random network features, resulting in linked data dependence. With this, the malware attack detection overhead using MLLD-MDQL is said to be reduced by 18% compared to [30] and 35% compared to [31].

3.1.3. Case 3: performance measure of false positive rate

In malware attack detection, when multiple comparisons are performed with respect to the FPR which is the probability for falsely rejecting the model to obtain the null hypothesis. The FPR has calculated the ratio among the number of negative events. The normal connections are identified wrongly are known as malware evaluated among the total number of actual negative events (i.e. total number of connections involving malware attack). This is mathematically expressed as given in (12).

$$FPR = \frac{FP}{FP+TN} \quad (12)$$

From (12), the false-positive rate ' FPR ', is measured based on the number of false positives ' FP ' and a number of true negatives ' TN ' respectively. It is measured in terms of percentage (%) shows the results of the false positive rate observed for three different methods, MLLD-MDQL, behavior-based malware detection [30] and malware based on behavior chains [31]. Finally, Figure 3 depicts the attack detection overhead for different connections. Hence, by applying different policies for each estimator, false positive rate is significantly reduced by 42% for the proposed method better when compared to the existing Behavior based malware detection model that obtained 62% of FPR.

3.2. Comparative analysis

The Table 2 shows the comparative analysis for the proposed MLLD-MDQL that is compared with the existing models. The performances are in terms of FPR as 12.85%, time as 25.52 ms, and overhead of 60-600 kB for the proposed method. However, the more detailed architectural design for a dedicated accelerator provided efficiencies better for chip area, power, and processing time was required to be investigated.

However, the developed model when comes under performances cost, it required still more improvement as cases signature were not enough.

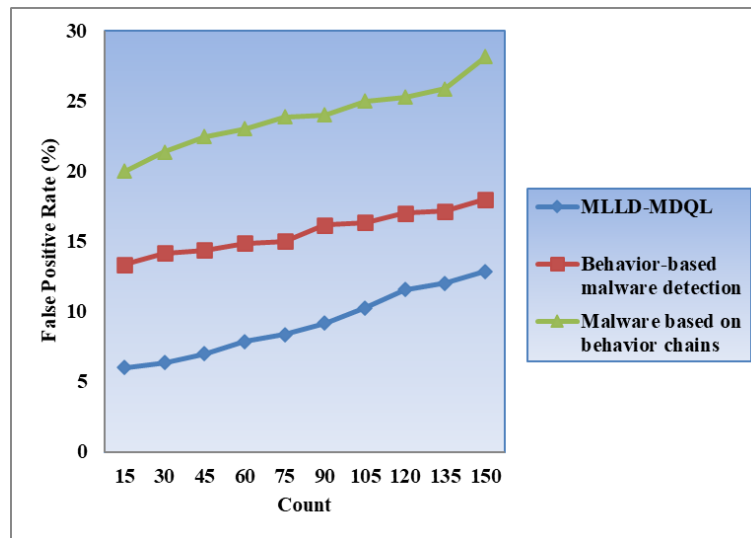


Figure 3. Attack detection overhead graphs with different connections

Table 2. Comparative analysis

Method	FPR (%)	Time	Overhead
Behavior-based malware detection [30]	3.17	-	-
Malware based on behavior chains [31]	-	90s	-
PMC-based malware detection [35]	12.5	6.825ms	-
MALDC [36]	-	20s	10%
Proposed MLLD-MDQL	12.85	25.525ms	60-600kB

4. CONCLUSION

The present research work discusses the new malware attack detection using the proposed MLLD-MDQL model. The main contributions of the proposed MLLD-MDQL method for malware attack detection reduce the malware attack detection time. The detection time is evaluated overhead and a false positive rate is involved. The proposed method reduces the attack detection time and overhead for operating the malware detection through multinomial linked latent dirichlet network behavior Extraction that extracts the network behavior patterns by monitoring API through data dependence. Next, with behavior grouping and classification is performed using modular double q learning malware classification, a false positive rate improves the results significantly. The simulations were conducted and the performance was evaluated to analyze malware attacks in terms of attack detection time, overhead and false positive rate. The results showed that the false positive rate is significantly reduced by 42% for the proposed method better when compared to the existing Behavior based malware detection model that obtained 62% of FPR. However, in the future Monitoring system events for malware prevention has to be determined.

REFERENCES





- [1] F. Xiao, Z. Lin, Y. Sun, and Y. Ma, "Malware detection based on deep learning of behavior graphs," *Mathematical Problems in Engineering*, vol. 2019, pp. 1–10, Feb. 2019, doi: 10.1155/2019/8195395.
- [2] N. Sharma and B. Arora, "Data mining and machine learning techniques for malware detection," in *Rising Threats in Expert Applications and Solutions. Advances in Intelligent Systems and Computing*, Singapore: Springer, 2020, pp. 557–567.
- [3] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, Dec. 2019, doi: 10.1186/s42400-019-0038-7.
- [4] A. O. Alzahrani and M. J. F. Alenazi, "Designing a network intrusion detection system based on machine learning for software defined networks," *Future Internet*, vol. 13, no. 5, p. 111, Apr. 2021, doi: 10.3390/fi13050111.
- [5] M. S. Abbasi, H. Al-Sahaf, M. Mansoori, and I. Welch, "Behavior-based ransomware classification: A particle swarm optimisation wrapper-based approach for feature selection," *Applied Soft Computing*, vol. 121, p. 108744, May 2022, doi: 10.1016/j.asoc.2022.108744.

- [6] G. Daniel, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, p. 102526, Mar. 2020, doi: 10.1016/j.jnca.2019.102526.
- [7] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
- [8] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on Autoencoders and API-images," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 26–33, Mar. 2020, doi: 10.1016/j.jpdc.2019.11.001.
- [9] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, Jan. 2020, doi: 10.1109/ACCESS.2019.2963724.
- [10] S. Yoo, S. Kim, S. Kim, and B. B. Kang, "AI-Hydra: advanced hybrid approach using random forest and deep learning for malware classification," *Information Sciences*, vol. 546, pp. 420–435, Feb. 2021, doi: 10.1016/j.ins.2020.08.082.
- [11] X. Gao, C. Hu, C. Shan, B. Liu, Z. Niu, and H. Xie, "Malware classification for the cloud via semi-supervised transfer learning," *Journal of Information Security and Applications*, vol. 55, p. 102661, Dec. 2020, doi: 10.1016/j.jisa.2020.102661.
- [12] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A Method for Windows Malware Detection Based on Deep Learning," *Journal of Signal Processing Systems*, vol. 93, no. 2–3, pp. 265–273, 2021, doi: 10.1007/s11265-020-01588-1.
- [13] V. Syrris and D. Geneiatakis, "On machine learning effectiveness for malware detection in Android OS using static analysis data," *Journal of Information Security and Applications*, vol. 59, p. 102794, Jun. 2021, doi: 10.1016/j.jisa.2021.102794.
- [14] T. Aishwarya and V. R. Kumar, "Machine learning and deep learning approaches to analyze and detect COVID-19: a review," *SN Computer Science*, vol. 2, no. 3, p. 226, May 2021, doi: 10.1007/s42979-021-00605-9.
- [15] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, p. 101663, Feb. 2020, doi: 10.1016/j.cose.2019.101663.
- [16] J. C. Olivares-Rojas, E. Reyes-Archundia, J. A. Gutiérrez-Gnecchi, I. Molina-Moreno, A. C. Téllez-Anguiano, and J. Cerda-Jacobo, "Smart metering system data analytics platform using multicore edge computing," *International Journal of Reconfigurable and Embedded Systems*, vol. 10, no. 1, pp. 11–17, 2021, doi: 10.11591/ijres.v10.i1.pp11-17.
- [17] D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," *Neural Computing and Applications*, vol. 31, no. 2, pp. 461–472, Feb. 2019, doi: 10.1007/s00521-017-3077-6.
- [18] K. Shirane, T. Yamamoto, and H. Tomiyama, "A design methodology for approximate multipliers in convolutional neural networks: A case of MNIST," *International Journal of Reconfigurable and Embedded Systems*, vol. 10, no. 1, pp. 1–10, Mar. 2021, doi: 10.11591/ijres.v10.i1.pp1-10.
- [19] L. A. Mohammed, T. A. Husain, and A. M. T. Ibraheem, "Implementation of SHE-PWM technique for single-phase inverter based on Arduino," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 4, pp. 2907–2915, Aug. 2021, doi: 10.11591/ijece.v11i4.pp2907-2915.
- [20] S. Gaddameedhi and P. Srinivas, "A novel fuzzy based controller to reduce circulating currents in parallel interleaved converter connected to the grid," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 2, pp. 1130–1142, Apr. 2021, doi: 10.11591/ijece.v11i2.pp1130-1142.
- [21] M. T. Yaqoob, M. K. Rahmat, S. M. M. Maharum, and M. M. Su'ud, "A Review on harmonics elimination in real time for cascaded H-bridge multilevel inverter using particle swarm optimization," *International Journal of Power Electronics and Drive Systems*, vol. 12, no. 1, pp. 228–240, Mar. 2021, doi: 10.11591/ijpeds.v12.i1.pp228-240.
- [22] H. Attia, H. S. Che, T. K. S. Freddy, and A. Elkhateb, "Mitigating the dead-time effects on harmonics spectrum of inverter waveform by the confined band VSPWM technique," *International Journal of Power Electronics and Drive Systems*, vol. 12, no. 1, pp. 295–303, Jun. 2021, doi: 10.11591/ijpeds.v12.i1.pp295-303.
- [23] A. Sharma, N. Anandh, and S. Gao, "Modulation index effect on inverter based induction motor drive," *International Journal of Power Electronics and Drive Systems*, vol. 11, no. 4, pp. 1785–1798, Dec. 2020, doi: 10.11591/ijpeds.v11.i4.pp1785-1798.
- [24] S. Ishak, S.-P. Koh, J.-D. Tan, S.-K. Tiong, and C.-P. Chen, "Corona fault detection in switchgear with extreme learning machine," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 2, pp. 558–564, Apr. 2020, doi: 10.11591/eei.v9i2.2058.
- [25] A. Hiendro, I. Yusuf, Junaidi, F. T. P. Wigiyanto, and Y. M. Simanjuntak, "Optimization of SHEPWM cascaded multilevel inverter switching patterns," *International Journal of Power Electronics and Drive Systems*, vol. 11, no. 3, pp. 1570–1578, Sep. 2020, doi: 10.11591/ijpeds.v11.i3.pp1570-1578.
- [26] T. Jing, A. Maklakov, A. Radionov, S. Baskov, and A. Kulmukhametova, "Research on hybrid SHEPWM based on different switching patterns," *International Journal of Power Electronics and Drive Systems*, vol. 10, no. 4, pp. 1875–1884, Dec. 2019, doi: 10.11591/ijpeds.v10.i4.pp1875-1884.
- [27] M. H. Yatim, A. Ponniran, and A. N. Kasiran, "Multilevel inverter with MPWM-LFT switching strategy for voltage THD minimization," *International Journal of Power Electronics and Drive Systems*, vol. 10, no. 3, pp. 1461–1468, Sep. 2019, doi: 10.11591/ijpeds.v10.i3.pp1461-1468.
- [28] M. Z. Aihsan, N. I. Ahmad, W. A. Mustafa, N. A. Rahman, and J. A. Soo, "Development of square wave inverter using DC/DC boost converter," *International Journal of Power Electronics and Drive Systems*, vol. 10, no. 2, pp. 636–644, Jun. 2019, doi: 10.11591/ijpeds.v10.i2.pp636-644.
- [29] M. Ahmed, M. K. Metwaly, and N. I. Elkalashy, "Performance investigation of multi-level inverter for DFIG during grid autoreclosure operation," *International Journal of Power Electronics and Drive Systems*, vol. 10, no. 1, pp. 454–462, Mar. 2019, doi: 10.11591/ijpeds.v10.i1.pp454-462.
- [30] J. Singh and J. Singh, "Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms," *Information and Software Technology*, vol. 121, p. 106273, May 2020, doi: 10.1016/j.infsof.2020.106273.
- [31] H. Zhang, W. Zhang, Z. Lv, A. K. Sangaiah, T. Huang, and N. Chilamkurti, "MALDC: a depth detection method for malware based on behavior chains," *World Wide Web*, vol. 23, no. 2, pp. 991–1010, Mar. 2019, doi: 10.1007/s11280-019-00675-z.
- [32] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96899–96911, May 2020, doi: 10.1109/access.2020.2995887.
- [33] Y. Zhao, B. Bo, Y. Feng, C. Xu, and B. Yu, "A feature extraction method of hybrid gram for malicious behavior based on machine learning," *Security and Communication Networks*, vol. 2019, pp. 1–8, Feb. 2019, doi: 10.1155/2019/2674684.
- [34] L. Liu, B. Wang, B. Yu, and Q. Zhong, "Automatic malware classification and new malware detection using machine learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 9, pp. 1336–1347, Sep. 2017, doi: 10.1631/FITEE.1601325.
- [35] J. Pattee, S. M. Anik, and B. K. Lee, "Performance monitoring counter based intelligent malware detection and design alternatives," *IEEE Access*, vol. 10, pp. 28685–28692, Mar. 2022, doi: 10.1109/ACCESS.2022.3157812.





- [36] M. Botacin, M. Z. Alves, D. Oliveira, and A. Grégio, "HEAVEN: a hardware-enhanced antivirus engine to accelerate real-time, signature-based malware detection," *Expert Systems with Applications*, vol. 201, p. 117083, Sep. 2022, doi: 10.1016/j.eswa.2022.117083.

BIOGRAPHIES OF AUTHORS



Sheelavathy Veerabhadrapa Kudrekar     pursued M. Tech. Degree in Computer Science & Engineering from JAIN University. She has 12 years of teaching experience. Her areas of interest and teaching include Big Data & Data Analytics, Data warehousing, Data Mining, Advanced Database management systems. She is pursuing her PhD. from REVA University. She has presented 1 paper in a national conference and published 1 paper in international journals. Qualification: B. E., M. Tech., (Ph.D.). She can be contacted at email: sheela.kv@reva.edu.in.



Udaya Rani Vinayakamurthy     received her Ph.D. from the Mother Teresa University, Kodaikanal, Tamil Nadu, India, in 2014, M.Tech. degree from MSRIT, Bangalore, India, in 2009 under VTU and B.E degree from SJCE, Mysore, India, in 1994 under VTU. She started her career as Lecturer in Govt. CPC Polytechnic, Mysore for one year. She served as Lecturer in various colleges during her career such as NIE- Mysore, SMSG JAIN College-Bangalore, RBANM's college- Bangalore, Nagarjuna College of Management- Bangalore for five years, as HOD & Assistant Professor in Sambhram Institute of Engineering- Bangalore for two years in the mean while she got BEST Lecturer Award. Now she is serving in REVA University as Senior Associate Professor from 2011. She is having a total of 22yrs of experience. She has published more than 60 journal papers in National and International Journals. Her research area includes Data Mining and Warehousing, machine Learning, Big data Analytics, Big Data and Hadoop, Genetic Algorithms. She is currently guiding 8 students through their research under VTU and REVA University. Currently she is working as a Senior Associate professor in School of C&IT, Head IPR, and as a PG coordinator in REVA University. She can be contacted at email: udayarani.v@reva.edu.in.