IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 29, No. 4, August, 2024, pp. 1591-1601
Journal homepage: http://iieta.org/journals/isi

Dynamic Malware Pattern Analysis with Rapid Node Behaviour Analysis Using Self Replication Model for Network Intrusion Detection

Ragini Mokkapati*, Venkata Lakshmi Dasari

School of Computer Science and Engineering, VIT-AP University, Amaravathi 522237, India

Corresponding Author Email: raginimokkapati@gmail.com

https://doi.org/10.18280/isi.290432

Received: 23 October 2023
Revised:21 March 2024
Accepted: 2 June 2024
Available online: 21 August 2024

Keywords:
intrusion detection, dynamic malware analysis, node behaviour, normal patterns, unusual patterns, network security

ABSTRACT

Advanced machine learning and artificial intelligence-based malware identification and categorization activities in real time are the primary emphasis of Malware Analysis and Intrusion identification in Cyber Physical Systems, along with the time sequence output of observed activity. Malware and other cyber threats have prompted the development of numerous static and behavior based detection approaches. These cyber security solutions show promise on large datasets, but they aren't reliable or resilient enough for real-world detection. Problems like virus detection and the identification of malevolent behavior highlight the critical need for improved cyber security solutions based on artificial intelligence. For those who utilize the internet, malware has become an enormous issue. The application is executed in a secure virtual environment and its actions are tracked in real-time to facilitate dynamic malware detection. A lot of people utilize API sequence analysis to find out if the software that is currently running is dangerous. While existing systems do consider API names and usage frequencies, feature mining of API sequence falls short, making it possible for some malware to evade detection. The two mainstays of dynamic analysis now in use either modify the virus itself or use an elevated component to execute the analysis. In contrast to the latter, which usually causes a discernible performance overhead, the former is instantly identifiable by even the most sophisticated malware. One of the most important steps in avoiding cyber assaults is developing new cyber security methods to detect hostile nodes before they communicate. Traditional dynamic malware detection models need to monitor the nodes more keenly for deep pattern analysis and eradicating nodes that cause malicious actions in the network. This research proposes a Dynamic Malware Pattern Analysis with Rapid Node Behaviour Analysis using Self Replication Model (DMPA-RNBA-SRM) for Network Intrusion Detection in the network. The normal patterns will be allowed into the network and the patterns of the nodes that are unusual are not allowed temporarily. The Pattern analysis and updating is performed and the detected patterns are analyzed and if they are malicious in nature, they will not be allowed into the network. The self replication model will be triggered when a unusual pattern is detected and required actions are performed in the network. The proposed model dynamic pattern analysis and detection is high when compared to traditional models.

1. INTRODUCTION

All computer-reliant networks and systems are increasingly vulnerable to malware attacks. The number and sophistication of newly released malicious software has increased dramatically in recent years, according to security researchers [1]. Malware caused over $1.8 trillion in losses in 2021, with more software vulnerabilities discovered than the year before, 2.2 million more malware signatures generated, 673 million more attacks reported [2]. Although half of all malware samples are merely repackaged versions of known malware, it is interesting to note that these samples still evade existing commercial-off-the-shelf antivirus software by using static signature-based techniques that are susceptible to code obfuscation and polymorphism [3]. In contrast, dynamic detection methods bypass such safeguards as code obfuscation and polymorphism by mimicking a process's behavior at runtime [4]. Typical dynamic techniques include system call scheduling and flow diagrams. Methods that rely on the sequence of system calls to develop a model of behavior. Four known issues with these approaches exist: the processing overhead of logging system calls; a substantial false alarm rate; the capacity to make a decision only after assessing the complete record of the processing procedure of a process and the ease that they can be bypassed by simply restructuring the system calls or adding insignificant system calls to invalidate the sequence that recursive methods rely on [5].

The intricacy and innovation of malware have grown throughout the years, despite the fact that it has existed since the early days of computing. Malicious software poses risks to private users, corporations, public services, governments, and security organizations; the most recent wave of ransomware has brought this issue into sharp focus. It is critical to identify malicious behavior as soon as possible, ideally before it does

harm, in order to safeguard these institutions and the public from malware attacks. But knowing what to search for can be challenging, particularly when dealing with previously unseen spyware.

Static and dynamic analysis methods, when applied to questionable files, can reveal useful information about the file's effect on the hosting system and, depending on the rules established by the approach, can assist establish if the file is harmful. Malware authors employ a variety of tactics to evade static analysis, such as code obfuscation, dynamic code loading, encryption, and packing. However, dynamic analysis can still uncover more about the files being analyzed, leading to improved detection capabilities. Although dynamic analysis outperforms static analysis, the methods and tools currently available for dynamic analysis have their limitations, and no one tool can account for every possible facet of malware behavior.

Software designed to carry out an attacker's malevolent aim is what we mean when we speak about malware detection [6]. Eliminating malicious software is a formidable challenge for researchers in the area of network security. As a result of poorly thought-out technological security countermeasures and software flaws, attackers are becoming increasingly adept at evading detection [7]. While antivirus software attempts to match incoming code with a database of known bad applications, NIDS [8, 9] examines network data and protocols at a granular level to detect suspicious or destructive activities. Because of its ability to hijack hosts, steal data, erase files, and disrupt network services, malware poses a significant threat to cyber security. Consequently, a great deal of research on malware analysis has focused on the host-based environment. Malware signatures are identifying patterns that malicious software leaves behind that antivirus software uses to protect users [10]. The amount of malware has grown substantially after obfuscation techniques were introduced and open source tools were widely available. Therefore, it is necessary to shift focus from identifying malware signatures and to study its behavior is required [11]. Malware detection relies heavily on being able to identify individual harmful characteristics. Malware detection will be faster and cheaper to maintain with this approach. The malware detection strategies are shown in Figure 1.

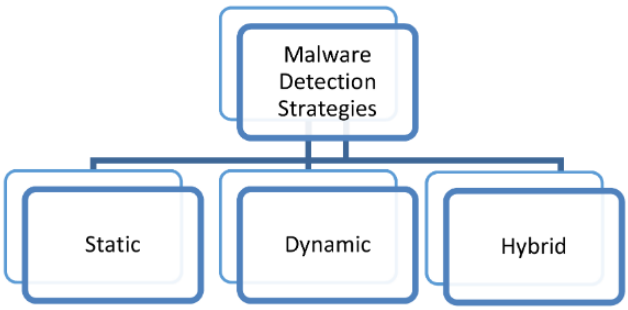

Figure 1. Malware detection models

Due to the complexity of analyzing infected machines, system logs, and malware files for indicators of compromise and potential goals, a reliable metric is needed for malicious characteristics detection [12]. Malware samples that have been compromised and classified by multiple anti-virus providers can be compiled on sites like VirusTotal [13]. However, the security sector has seen that several antivirus vendors employ various malware labels [14]. Using supplementary tools, such as on-demand detection methods like function call analysis, may show the malicious code's execution behavior [15]. For instance, they provide entry to the Application Programming Interface (API) and the system resources it provides. Previous studies have shown that malicious code often operates by carrying out a predetermined list of commands [16]. If a piece of code produces and writes files, for instance, that could be interpreted as malevolent behavior. The NIDS model is shown in Figure 2.

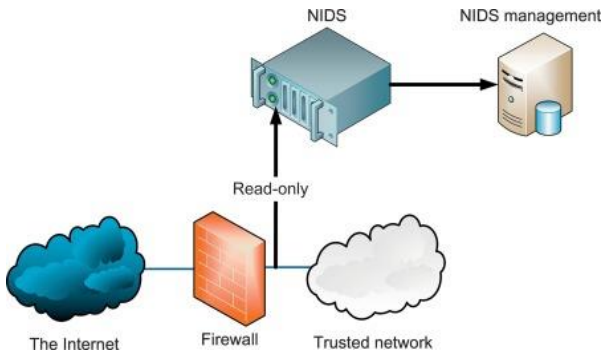

Figure 2. NIDS model

Devices inserted during scenario creation are just one example of the new cyber risks brought about by the lightning-fast development of networking, computing, sensing, and control systems. More and more, contemporary vehicles, medical equipment, and smart industrial systems are breaking down due to external hazards and inefficient operations. Turning on the system's built-in self-healing feature is essential for maintaining service quality [17]. We want to conduct a state-of-the-art analysis to identify the areas where cyber-physical systems could benefit from self-healing through machine learning, with the aim of improving security and preventing breakdowns [18]. In this study, the three main components of computer systems' self-healing capabilities—fault warning, dynamic malware detection, and fault auto-remediation—are outlined. These factors must be carefully considered in order to incorporate self-healing features. To fully benefit from these characteristics, one must be familiar with the underlying self-healing theories [19]. Cyber-physical system self-healing capabilities promise to be an exciting new research topic for the creation of cutting-edge IT. Self-organization and repair capabilities in cyber-physical systems could improve system security and user satisfaction [20]. A self-healing technology deployed on a power grid, for example, would repair damage or threats automatically, allowing communities to continue receiving power and vital services to remain operational despite disruptions [21].

As digital technologies become increasingly integrated into many sectors of the global economy, including trade, manufacturing, healthcare, and government, so too will the number and sophistication of attacks on computer systems and networks. One aspect of this risk is cyber attacks, both individual and governmental [22]. They targeted anyone who stayed indoors during the COVID-19 pandemic lockdowns, as well as schools, businesses, hospitals, and other places of employment [23]. Research into self-healing computer systems has become increasingly significant in recent years as broad use of digital technology has left communities more susceptible to damaging cyber-attacks [24]. Cyber-physical systems must be protected from intruders, and this has prompted an examination of the most up-to-date approaches to

this problem [25]. Industry 4.0 equipment, which includes cyber-physical systems, are Internet-enabled upgrades from the earlier, less intelligent gadgets of Industry 3.0. One example of the shift from manual repairs to self-repairing software used to safeguard cutting-edge systems nowadays [26]. There is a shift toward automating the protection process via self-healing in networks in an effort to cut expenses. The self healing model in networks is shown in Figure 3.

Latency, bandwidth, and scalability issues plague the majority of self-healing system installations. In order to improve network security and address issues, this study proposes a standardized architecture for dynamic malware detection and a self-healing model [27]. Innovative cyber security systems are necessary to identify hostile nodes before they interact, which is a crucial step in averting cyber assaults [28]. Traditional dynamic malware detection approaches necessitate closer node monitoring in order to exclude nodes causing harmful network activity through in-depth pattern analysis. Dynamic Malware Pattern Analysis with Rapid Node Behaviour Analysis utilizing Self Replication Model is proposed in this research for Network Intrusion Detection. The regular patterns will be admitted to the network, while the abnormal patterns of the nodes will be blocked for the time being.

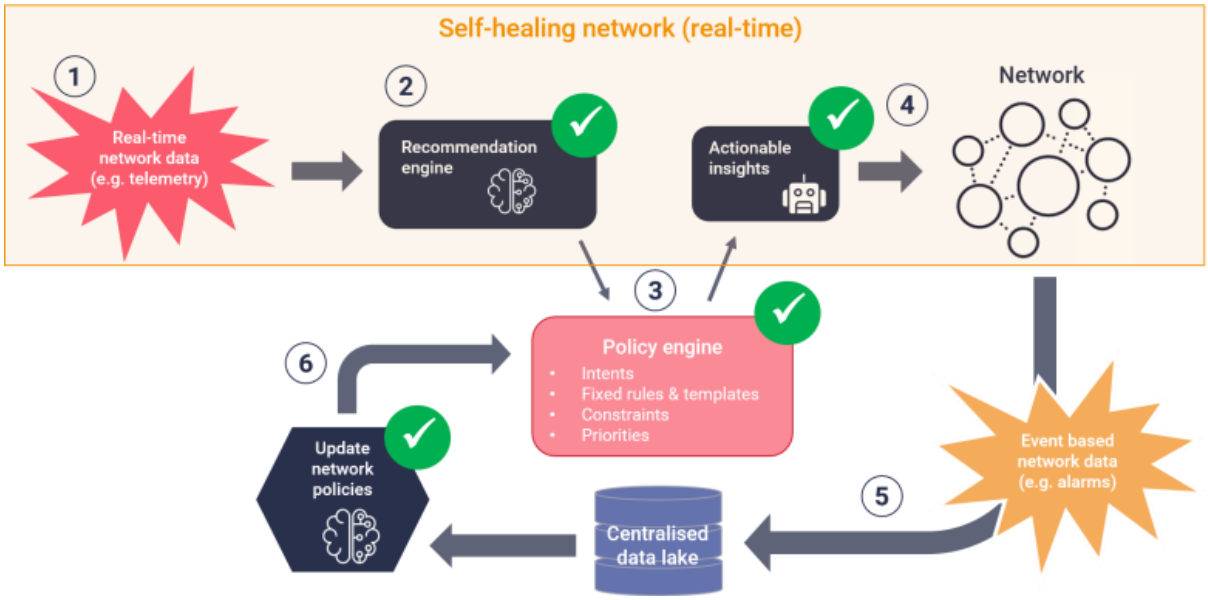

Figure 3. Self healing model

2. LITERATURE SURVEY

With the help of the IoT, we can pave the way for a world where everything is online and can share information in real time. When combined, the IoT, 5G, and AI open the door to smart city and smart factory applications. Threats to the security of IoT infrastructure, applications, and devices have multiplied in tandem with the IoT's stratospheric ascent in popularity. There have been several lines of inquiry on the best ways to safeguard Internet of Things devices from malicious malware. The dynamic nature of IoT malware makes it difficult to keep up with static analysis, even though it can rely on current models to detect harmful IoT code. An intelligent method for detecting common and unique IoT malware, called dynamic analysis for IoT malware detection (DAIMD), was presented by Jeon et al. [1] to safeguard IoT devices from harm. In a layered cloud environment, DAIMD can use a convolutional neural network (CNN) model to learn about IoT malware and do dynamic assessments. The dynamic analysis of DAIMD in a nested cloud environment can be used to extract features of IoT malware from memory, networks, virtual file systems, processes, and system calls. The CNN is employed to classify and train the behavior images of IoT malware by transforming the retrieved and processed behavior data into visuals. In order to lessen the impact of malware assaults on Internet of Things devices, DAIMD analyzes and learns from data on the behaviors that these devices dynamically generate.

Malware is becoming an increasingly serious threat to computers in the current era. Code obfuscations are only one example of how malware writers use a plethora of complex features to build new malware and evade current malware detection techniques. Classifying new malware variants with similar characteristics into the right families is still quite difficult, even when the classifier is trained with known variants from the same family. One of the challenges in establishing a more universal method for identifying malware is recognizing and extracting features that are distinctive to each type of malware. The diversity of malware makes it challenging to design features that improve the classifier's generalizability. Due to their reliance on static signature-based techniques or dynamic behavior-based methodologies, traditional malware detection technologies are incapable of analyzing and detecting advanced and zero-day malware. By using a visualization method where malware is represented as 2D visuals, Roseline et al. [2] offered a strong machine learning-based anti-malware solution to these problems. In order to achieve better results than deep learning methods, the suggested system is built on a multilayered ensemble approach. It is possible to use the proposed method with reduced model complexity and avoid hyperparameter adjustment and backpropagation.

Although smartphones were Android's original intended users, the platform has already spread to various kinds of mobile devices. There has been a meteoric rise in the production of varied, pervasive, and complex Android

malware due to the platform's open architecture and APIs, which allow for the hosting of third-party programs. Bibi et al. [3] demonstrated an Android method for multi-class malware detection using Cuda and the Gated Recurrent Unit (GRU). To test the efficacy of the method, it was applied to Android apps drawn from cutting-edge datasets. Accuracy, precision, recall, F1-score, and other standard performance evaluation measures were used in conjunction with our own DL-driven architectures and test procedures to thoroughly analyze the suggested method's performance. Despite a small decrease in speed efficiency, the GRU-based malware detection system excels at identifying malware, with a detection rate of 98.99%.

A billion dollars' worth of demand has resulted from the technological revolution that smart devices powered by Android have brought about in raising people's standard of living. Even if the Android market is all the rage, the widespread and potentially dangerous mobile virus has made the popular Android platform a prime target for cybercriminals. Because it may be constructed on any executable file extension and imitate real third-party programs, multivector malware is extremely difficult to detect. Haq et al. [4] presented an intelligent multi-vector malware detection mechanism that is very effective and powered by hybrid deep learning (DL). This mechanism simplifies a complex and crucial matter. The approach used Bidirectional Long Short-Term Memory (BiLSTM) and Convolutional Neural Networks (CNNs) to efficiently detect chronic infections. Not only were state-of-the-art hybrid DL-driven architectures and benchmark DL algorithms used to assess the efficacy of the proposed technique, but so were publicly available datasets and standard performance indicators. Additionally, cross-validation proves that the suggested framework is efficient in detecting irregularities quickly and accurately.

As an integral part of cyber-physical-social systems, computing at the cloud's edge is gaining prominence in many parts of the globe. When it comes to cloud computing, virtualization is a game-changer. It lets us quickly deploy computing workloads by transferring virtual machines (VMs), hence cloud security is super important. In an infrastructure as a service (IaaS) environment, Gan et al. [5] presented a way to stop malware from spreading among virtual machines (VMs). Initially, a dynamical propagation model was used to examine the impact of antivirus software on virtual machines (VMs) and other important elements involved in the spread of malware. To get insight into the spread of malware in an infected cloud environment, the author conducted a theoretical analysis of the model using differential dynamics. To test the strategy's viability and efficacy, several numerical simulations are run.

The massive volume of industrial IoT malware, which includes new kinds of advanced persistent threats and forms of obfuscation, is one of the most critical security challenges in today's linked world. A powerful platform for detecting Android malware apps in IIoT, Fed-IIoT was introduced by Taheri et al. [6]. The participant side generates data due to two dynamic poisoning attacks, one using a generative adversarial network (GAN) and the other a federated GAN. On the server side, the author keeps an eye on the global model and shapes it into a robust collaboration training model by avoiding anomalies in aggregation using a GAN network (A3GAN) and refining two GAN-based countermeasure algorithms. One major benefit of Fed-IIoT is that it enables devices to discreetly and securely communicate with one other as part of the IIoT. The author conducted tests on three separate IoT datasets, each with its own unique set of characteristics.

Astrocytes are endogenous cannabinoid retrograde messengers that have been demonstrated in many studies to connect with neuronal presynaptic receptors and postsynaptic spines. A circuit for self-repairing neuron networks based on memristors was introduced by Hong et al. [7]. This circuit mimics the biological self-repair mechanism of astrocytes by mimicking changes in neurotransmitters at a given threshold. The following components make up the proposed circuit, which collectively mimic the brain's astrocyte-neuron network: This is broken down into four sections: One module increases the likelihood of synaptic information release in neurons; another module detects network problems; a fourth module increases the likelihood of release in the astrocyte-neuron circuit. When the PR is low due to a malfunctioning synapse, the neuron module is practically silent. An automatic malfunction detector is built into the circuit. A process similar to that carried out physiologically by astrocytes can be used to restore injured neurons by increasing the PR of nearby healthy neurons. The broken circuit can be repaired using this technique. As the virtual circuit showed, a circuit's self-healing capacity grows in proportion to the number of neurons it contains. However, when an astrocyte-neuron network has more injured neurons, its capacity to heal itself decreases, and its performance plummets. The robots' functionality and dependability were greatly enhanced by the self-repairing circuit.

With the dual objectives of rectifying the logical network topology and sustaining synchronization within the physical mobile robot formation, Liu et al. [8] took a robotics-centric approach to the self-repair problem in their study. The authors propose a self-repairing method based on gradients that relies solely on local interactions between coupling robots. Incorporating a method for gradient production and diffusion that can consistently disperse gradients throughout the robot formation is the first stage in achieving decentralized optimization of the repair path. Following an introduction to the recursive self-repairing approach and the suggested gradient distribution, the self-repairing problem is demonstrated to be solved by means of a number of self-repairing rules, each with its own associated control mechanism. The approach has been tested and proven to enhance robot formation coordination and repair path quality through both theoretical and practical measurements. Results from both theoretical and practical investigations corroborate the viability and efficiency of the suggested approach.

The introduction of vehicle ad hoc networks (VANETs) has allowed for an increasing diversity of automobile applications. Throughput and multiplexing of routing routes are usually more important to applications than delay and effective forwarding rate. It is difficult to establish dependable multihop forwarding circuits in VANETs in real time because of factors like inconsistent topological information, unconnected networks, high churn rates, etc. In their proposal of an ant colony optimization (ACO) based self-healing routing strategy (SR), Liu et al. [9] aimed to assist VANETs in keeping up with the latest developments in research and development. An innovative feature of SR, the ACO algorithm finds the optimal routes in terms of latency and connectivity. Everyone agrees that the RBA is the best way to measure transportation efficiency. Fuzzy logic systems could be useful in determining the RBA from the PDR and delay, which could be computationally expensive. If you want to increase throughput, lengthen the optimum path, and reduce the time and effort

needed to reestablish a path after a break in its continuity, this article advises fixing at crossings and in the road.

An adaptive smart building that can repair itself in response to external stimuli has a two-pronged approach to energy management, as detailed by Akbari-Dibavar et al. [10]. In addition to connecting to the city's primary power grid, the state-of-the-art building also features its own diesel generator and photovoltaic panels. Software for managing energy use ought to be intelligent enough to allocate resources strategically. Bilevel programming starts by using MTR predictions to prioritise component failures. First, we use stochastic programming to account for the uncertainty of faults. Then, we use robust optimization to deal with the unpredictability of real-time market prices. After significant problems have been identified, we use this two-stage hybrid robust-stochastic software technique to figure out the bid/offer in both the day-ahead and actual time energy markets. Ultimately, GAMS's optimization model is mixed-integer linear programming after additional linearization.

3. PROPOSED MODEL

By identifying the node's actions, users can ensure that the communication channels are being constantly monitored. It will aid in identifying the malicious nodes that are impeding communication. The primary goal is to identify the nodes that are the root of the packet loss problem and eliminate them. Congestion refers to the total breakdown in communication that selfish nodes might generate [29]. Disruptions to the network can occur when individual nodes alter their behavior. To stop it, just take out the bad actor from the network. Because of this, neighboring nodes will be slammed with more work. An unstable and underperforming network is the direct effect of the disconnected node [30]. By identifying the malicious nodes, the network can gain confidence with behavioral prediction. This prediction method's key benefit is that it can identify malevolent nodes from selfish ones and kick them off the network if they are the source of transmission failures [31].

Quality of Service (QoS) is becoming more important to service providers as self-healing functions enable the network environment to recover from failure scenarios that may occur within software, networks, or hardware components of the system. Overload, intrusion, misconfiguration, and other complex failures in various parts of the network are the root cause of the problem that hinders the network's correct performance. Recent studies have shown that systems with built-in self-healing capabilities can reduce the societal and economic impact of these breakdowns [32]. The integration of a fault-solving method library, a twin model system, and a deep learning (DL) algorithm into a network to create a self-healing mechanism is one of several proposed solutions. In order to implement intrusion detection in the cyber-physical system, the DL algorithm must be built using datasets obtained from the fault-solving library and thereafter deployed. Traditional malware protection relies on scanning specified areas for known virus signatures; to incorporate self-healing capabilities, one must first undergo intrusion detection. Once this is complete, the fault classification process can begin, and the fault-solving library can provide a workable mitigation solution.

Despite the fact that the conventional method is necessary for safeguarding against known threats, it does not provide any defense against zero-day vulnerabilities, which are brand-new attacks for which no security patch has yet been provided. Their signature has not yet been recognized by anti-virus programs, thus the nature of the attack is yet unknown. Because of their inability to learn and adapt on their own, such systems rely on periodic external upgrades to keep their distinctive knowledge base current. One of the most pressing questions in cyber security is whether or not a system can proactively identify and respond to previously unknown threats, thereby limiting any potential damage and, ideally, eliminating them altogether. For this reason, a wide variety of AI procedures, such as anomaly detection, are frequently used. Both the false-negative and false-positive error rates of the detection techniques should be kept to a minimum. Existing attacks can cause serious harm to the system if they go undetected. False positives occur when a legitimate action is incorrectly labeled as malicious by the system, prompting countermeasures that can do as much financial and operational damage as an undetected attack.

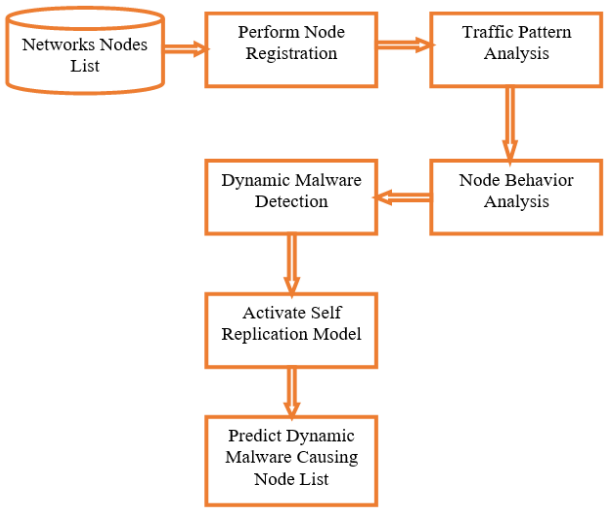

Figure 4. Proposed model framework

In addition to detecting and blocking attacks, a self-healing system may fix any harm that has been done. Finding intrusions using dynamic patterns is the goal of this research, which employs pattern recognition and intrusion detection. Intrusions are detected, sequence patterns are formed, and environmental events are monitored. The algorithm checks for a correlation between past behaviors and potentially harmful environmental signs when they are discovered. Instances of a system are alerted and the system's defenses are activated when they notice a sequence of potentially dangerous events. Commonly, "non-self detection" describes the process of finding intrusions in negative selection. This is due to the fact that the algorithm shares many similarities with how a live thing distinguishes its own cells from those of an outside source. To create non-self entities that cannot be separated from self entities using similarity testing is the core idea. A new entity is considered foreign to the system when it is compared to one of these non-self entities. The proposed model framework is shown in Figure 4.

Since it is possible to run each detector comparison on its own host, the detection approach is inherently portable. If hosts so desire, they can even employ their own sensors. Since negative selection is mainly an intrusion detection approach, it can be used even if no prior knowledge of intrusions is known.

A network of detectors is self-learning because it will evolve over time when some no longer function and new ones are added depending on the traffic of recent events. It is possible to monitor self-entities in order to track detector changes and vice versa. Reliable computers have communicated with one another to issue a warning. Every system in the network is alerted whenever a single machine detects and identifies an attack. They check their pasts to see whether they've encountered something similar or if a certain prefix of the sequence of events seems familiar, since an attack might still be in its early stages. This makes sense because the same malicious pattern could originate from different places or employ several variables on each computer, increasing the diversity of events that could be matched. When one machine's sequence only partially matches another's sequence, the system should advise administrators of a potential hazard but keep executing all other orders to prevent overreaction. If there is a database like this, it may be used to compare the updated timeline to a potentially harmful sequence. As soon as any potentially harmful sequence occurs inside that time frame, a protective mechanism is initiated. Normal communication is maintained in the absence of a match. This research proposes a Dynamic Malware Pattern Analysis with Rapid Node Behaviour Analysis using Self Replication Model (DMPA-RNBA-SRM) for Network Intrusion Detection in the network. The normal patterns will be allowed into the network and the patterns of the nodes that are unusual are not allowed temporarily. The Pattern analysis and updating is performed and the detected patterns are analyzed and if they are malicious in nature, they will not be allowed into the network.

Initially consider the network that contains nodes as a list $\{N_1,N_2,\ldots\ldots..,N_K\}$ where k is the total number of nodes in the WSN. The nodes that are considered in the network will be allocated with a label of values that are used for node authentication and recognition. This label helps in detection of each node in the network. The node label allocation is performed as:

$$Tconst[K] = \sum_{i=1}^{K} getCurrentTime(i)$$

$$\omega[K] = \sum_{i=1}^{K} getnodeattr(i) + noderange(i)$$

$$Nlabel[K] = \sum_{i=1}^{K} \frac{\sum_{i=1}^{k} getnodeaddr(i)}{K} + Tcont(i) + \omega(i) + rand(i)$$

Here Tconst is the time constant captured at every node registration, ω is the node properties and rand() is used to generate a random number for each node at label allocation.

Consider a traffic pattern set $\{TP_1,TP_2,\ldots\ldots TP_N\}$ where there can be N traffic frames to communicate. Each traffic pattern is analyzed for detection of change in the network traffic data. The traffic pattern analysis is performed as:

$$TraData[K] = \prod_{i=1}^{K} \frac{\sum_{i=1}^{k} getattr(i) + Nlabel(i)}{K} + \max\left(range\big(node(i)\big)\right) + \min\left(range\big(node(i)\big)\right)$$

$$PatternAnaly[K] = \sum_{i=1}^{K} \max(TraData(TP(i))) - \min\big(TraData(TP(i))\big) + \lim_{i\to K}\left(\mu(i) + \frac{\beta(i)}{K-1}\right)^2$$

Here μ represents the normal sequences detected in data analyzed in the nodes that are authorized. β represents the changes in the pattern sequences in the data frames.

The node behaviour is analyzed for changes in the network traffic. The node behaviour represents the actions in the network. The node behaviour analysis is performed as:

$$NodeBehaviour[K] = \sum_{i=1}^{K} \frac{nodeattr(i)}{K} + simm\big(\beta(i),\beta(i+1)\big) - \min\left(range\big(node(i)\big)\right) + Nlabel(i)$$

$$NStatus[K] = \sum_{i=1}^{K} \delta\big(NodeBehaviour(i,i+1)\big) + \max\left(TP(i,i+1)\right) + range(\beta(i)) \begin{cases} NStatus \leftarrow 1 & if\ (\beta(i) > Th) \\ NStatus \leftarrow 0 & Otherwise \end{cases}$$

Here δ is the status of the node behaviour among the nodes and the Th represents the threshold limit of changes in the similarity.

With a dynamic investigation, previously unseen malware can be found. By executing it on a network system, IT security experts can observe malware's activity without worrying about infecting their own networks or systems. This type of analysis is crucial in a world where malware is always evolving because it gives full visibility into the danger and what drives it. Let dynamic malware patterns set $\{MW_1,MW_2,\ldots\ldots MW_P\}$ where p is the total malware patterns detected.

$$MalPat[K] = \sum_{i=1}^{K} getrange(MW_i) + \max\left(NStatus(i,i+1)\right)$$

Dynamic malware prediction is performed by analyzing each node traffic patterns and the changes in the patterns are detected. The nodes which are causing the malicious actions in the network are identified and then the nodes will be removed using the self healing model. The dynamic malware pattern analysis is performed as

$$DynMal[K] = \sum_{i=1}^{K} \frac{\sum_{i=1}^{K} getNStatus(i)}{len(K)} + \max(MW_i, MW_{i+1}) + \frac{Max(MalPat(i,i+1))}{len(K)} simm(NStatus\big(node(i,i+1)\big))$$

In a self-healing network, problems are resolved automatically, without the need for human involvement. A network automation tool can detect and repair any kind of network issue or security violation in real time. While there

are many possible implementations of self-healing networks, the vast majority of them include receiving an alarm that causes your network to take corrective action. The malware is entirely eradicated from the network once the self-healing model examines every pattern and fixes it. The self healing model operations and the nodes causing the malicious actions are listed as:

$$SelfHeal[K] = \prod_{i=1}^{K} \frac{Nlabel(i)}{K} + \tau\left(rm\left(DynMal(node(i))\right)\right) + rm(MalPat(i) - node(Nlabel(i))$$

$$MalAlarm[K] = \sum_{i=1}^{K} \frac{\sum_{i=1}^{K} diff(DynMal(i, i+1))}{len(K)} + \gamma(simm(MalPat(i.i+1)))$$

$$SelfRelica[K] = \sum_{i=1}^{K} \prod_{i=1}^{K} getmaxrange(DynMal(i, i+1)) + activate(SelfHeal(i)) + \min\left(simm(DynMal(i, i+1)\right) - \max\left(simm(DynMal(i, i+1))\right) + maxAlarm(node(i))$$

Here τ is the model that generates the copies of the code in the network systems, rm is used to remove the specified node that has malware patterns, γ is the models for triggering the alarm for self replication model to activate self healing model.

4. RESULTS

Wireless sensor networks differ from their wired counterparts in that they consist of a dispersed collection of motes that possess the self-organizing quality. Because of their unique architecture, intruder detection algorithms for wireless sensor networks must account for the limited processing capacity and high energy consumption of the former. Due to their extremely high cost per mole, localization-based approaches for intrusion detection employing pattern analysis are not practical. There are often a few distinct types of intrusion detection systems.The public signature method of misuse detection uses low resources but is susceptible to attacks that have not been made public. Intrusion detection can find both known and undiscovered hazards by comparing the subject's typical behavior with its reported activity. If the nodes conduct is inconsistent with the estimated or projected one, the system must issue a report. The log and audit files of some models may need tweaking before they can be used with wireless sensor networks to their full potential.

Behavioral prediction provides a generic paradigm for predicting nodes' actions based on their routing. Behavior probability estimation is a tool that can be used to boost network confidence. Trustworthy category nodes can be formed to help identify malicious nodes and to build confidence between nodes that are far away. Remember that network traffic can interrupt connections while keeping an eye on the pattern of activities. Conventional communication networks rely on reliable routing algorithms to ensure node resilience. For WSN routing to work, faraway nodes must prove their reliability. Managing the forward node is critical to completing the routing procedure in this WSN. The pattern of activity in a WSN is determined by analyzing the process of node packet delivery. The main reason to use this technique is the ability to control node development at the location. This method uncovers harmful attacks and builds trust between distant nodes. In the event that an unreliable node is identified, the reliable ones will need to adjust their configurations. If a node's harmful actions impact the entire network, we call it a failure node. How every node will behave is hard to predict. Returning the system to normalcy requires early detection of the rogue node. But if the node is egocentric, then it needs to be on its own. This research proposes a Dynamic Malware Pattern Analysis with Rapid Node Behaviour Analysis using Self Replication Model (DMPA-RNBA-SRM) for Network Intrusion Detection in the network. The normal patterns will be allowed into the network and the patterns of the nodes that are unusual. The proposed model is compared with the traditional Dynamic Analysis for IoT Malware Detection with Convolution Neural Network (DA-IoT-MDCNN) Model and Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm (IVMD-RFP).

The proposed model is implemented in python and executed in Google Colab. The proposed model considers the dataset from the link https://www.kaggle.com/c/malware-detection/data. The system configurations are Intel I5 processor, 8GB RAM, 1TB HDD, 3.70 GHz. I created this dataset as an outcome of my research into android security and machine learning. The information was gathered by a procedure that included making a binary vector of permissions utilized for every application examined, where 1 indicates usage and 0 indicates non-usage. In addition, the malware/benign samples were categorized by "Type"; one sample was classified as malware and the other was not.

The proposed model performs gathering of complete node information for all the nodes in the network. The nodes information is maintained for node identification and future communications with the nodes. The Node Registration Time Levels of the proposed and traditional models are shown in Table 1 and Figure 5.

When a link, file, or connection is made or acquired over the network that raises suspicion, it is considered malware traffic. Malicious traffic is a security risk that can compromise computer or the security of a organization. There are several methods and technologies available for detecting malicious network activity. Monitoring network activity for suspicious patterns, abnormalities, or known signatures of threats is where IDS and IPS come in handy. The Table 2 and Figure 6 represents the Traffic Pattern Analysis Accuracy Levels of the proposed and traditional models.

Table 1. Node registrations

Nodes Considered in the Network	Models Considered		
	DMPA-RNBA-SRM Model	DA-IoT-MDCNN Model	IVMD-RFP Model
50	10.3	14.7	19.1
100	10.7	15.2	19.4
150	11	15.8	19.7
200	11.3	16.3	20.1
250	11.7	16.7	20.5
300	12	17	21

Table 2. Traffic pattern analysis

Nodes Considered in the Network	Models Considered		
	DMPA-RNBA-SRM Model	DA-IoT-MDCNN Model	IVMD-RFP Model
50	97.6	91.5	93.7
100	97.8	92	94
150	98	92.5	94.2
200	98.1	93	94.5
250	98.3	93.5	94.6
300	98.4	94	95

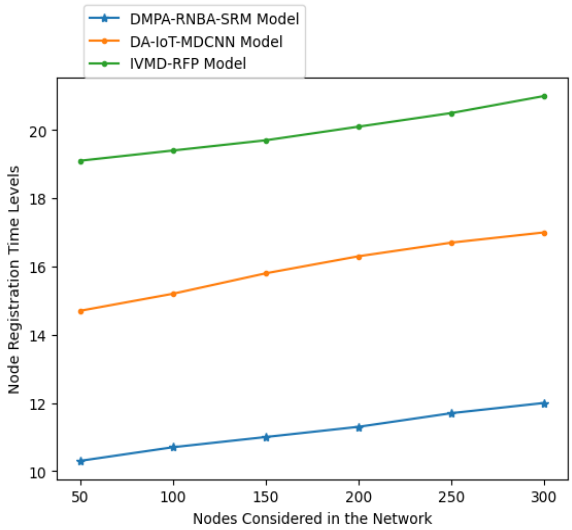

Figure 5. Node registration time levels

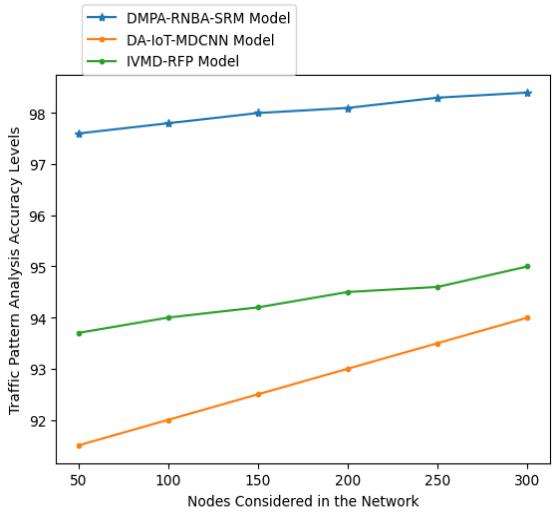

Figure 6. Traffic pattern analysis accuracy levels

Table 3. Node behavior analysis

Nodes Considered in the Network	Models Considered		
	DMPA-RNBA-SRM Model	DA-IoT-MDCNN Model	IVMD-RFP Model
50	97.6	94.2	91
100	97.9	94.5	91.2
150	98	94.7	91.4
200	98.2	95	91.6
250	98.4	95.3	91.8
300	98.5	95.5	92

Table 4. Dynamic malware detection

Nodes Considered in the Network	Models Considered		
	DMPA-RNBA-SRM Model	DA-IoT-MDCNN Model	IVMD-RFP Model
50	97.6	93.7	92.2
100	97.8	94	92.6
150	98	94.2	92.8
200	98.2	94.5	93
250	98.4	94.7	93.3
300	98.6	95	93.5

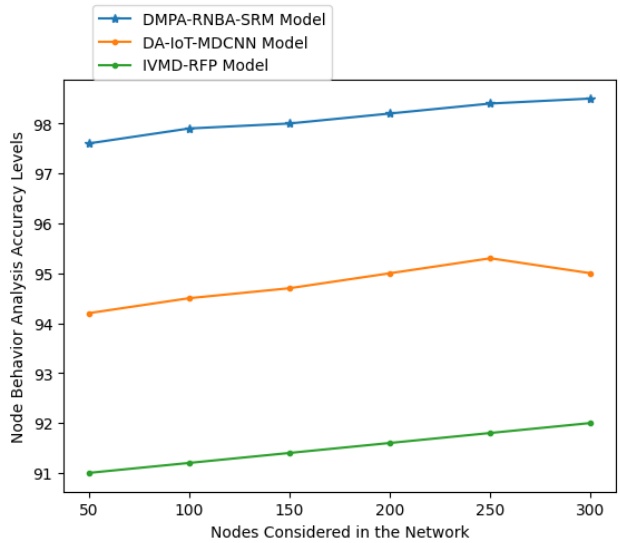

Figure 7. Node behavior analysis accuracy levels

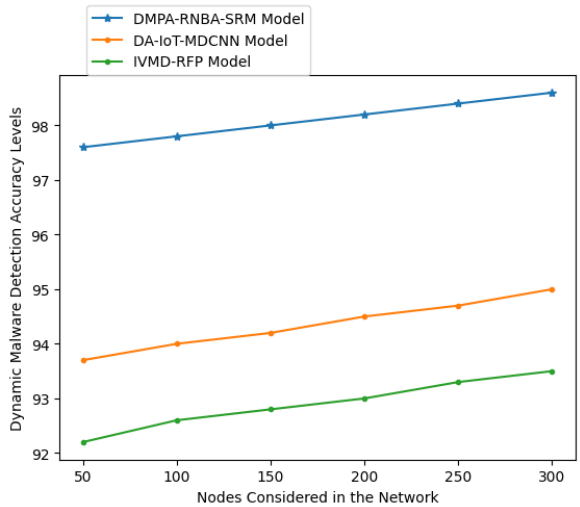

Figure 8. Dynamic malware detection accuracy levels

Each node in the network has its own behaviour. The behaviour of each node is analyzed to recognize the node actions in the network. The change in node behaviour results in cause of malicious actions. If any node is affected with malware, such nodes will cause malicious actions in the network. The Table 3 and Figure 7 depict the Node Behavior Analysis Accuracy Levels of the existing and proposed models.

When performing dynamic malware analysis, any suspicious code is run in a network to ensure it does not cause harm. There is no chance of the malware infecting the machine or escaping into the network because the system is closed.

Observing and recording the program's behavior both within and outside of the network is crucial for dynamic malware analysis, which takes a node behavior based method to understanding possible dangers. The Dynamic Malware Detection Accuracy Levels of the proposed and existing models are shown in Table 4 and Figure 8.

The proposed self-healing theories aims to provide an explanation of the basic concepts that must be considered while implementing self-healing functionality, as well as the pattern between self-healing and other scientific disciplines. What it means for a cyber-physical system to have self-healing functionality built into it, while the methods for achieving self-healing include specifics about the underlying models, frameworks, and network topologies. The self healing model removes the detected malware in the network and removes the malware patterns for increasing the network efficiency. The Self Replication Model Performance Accuracy Levels of the proposed and existing models are shown in Table 5 and Figure 9.

Table 5. Self replication performance

Nodes Considered in the Network	Models Considered		
	DMPA-RNBA-SRM Model	DA-IoT-MDCNN Model	IVMD-RFP Model
50	97.3	93.5	92.7
100	97.5	94	93
150	97.8	94.5	93.2
200	98	95	93.4
250	98.2	95.5	93.6
300	98.4	96	94

Table 6. Network security

Nodes Considered in the Network	Models Considered		
	DMPA-RNBA-SRM Model	DA-IoT-MDCNN Model	IVMD-RFP Model
50	97.9	93	92.4
100	98	93.2	92.8
150	98.1	93.4	93
200	98.3	93.7	94.3
250	98.4	93.8	94.6
300	98.6	94	95

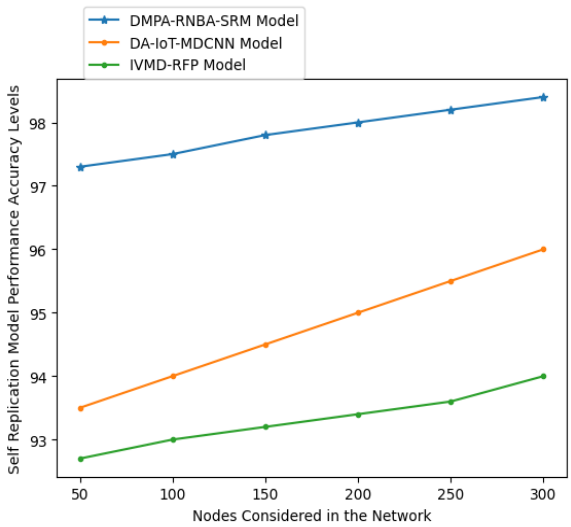

Figure 9. Self replication model performance accuracy levels

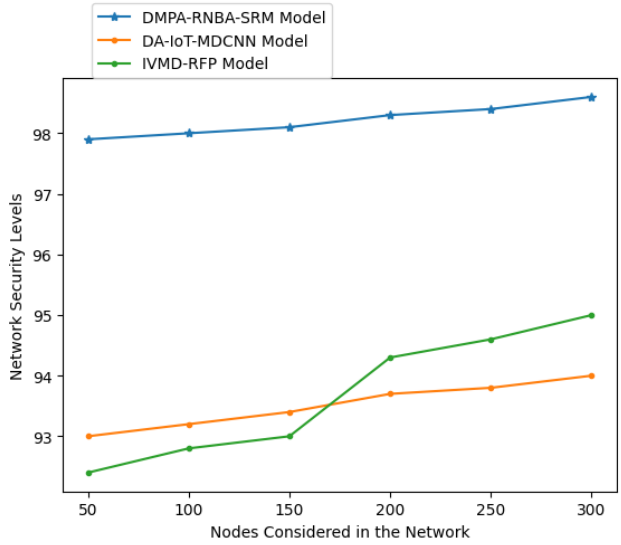

Figure 10. Network security levels

To secure the network, one must take any and all measures necessary to keep data and network operational at all times. Hardware and software are both part of this. It eliminates multiple dangers at once. It prevents them from getting into ones system or spreading throughout it. Accurate network security controls who can access the network and how. The network security levels of the existing and proposed model are shown in Table 6 and Figure 10.

5. CONCLUSION

To conduct a dynamic malware analysis, a secure environment is used to execute hazardous code while meticulously monitoring its actions to assess their level of impact. Dynamic malware analysis allows malware to unpack and operate in a network scenario, sidestepping the constraints of static analysis. Running an untested program and seeing what happens is the goal, but there are major downsides to this seemingly simple approach. The malware could do permanent damage to the computer and its data if left running. The application could distribute viruses of any kind and cause network congestion. There is not a single thing about this that would be uplifting. A network separates the infected program from the rest of the system, protecting it from potential harm. In its most basic definition, malicious software is any program that is installed on a computer without the owner's consent or knowledge and has the purpose of causing harm to the host's data, apps, or operating system. Malware usually uses sneaky or illegal means to spread itself through files in order to infect a user's system, which is the initial step for infecting a computer. Tales of cyber assaults often make mention of malware families such bots, viruses, spyware, worms, scareware, and trojans. A self-replicating model for network intrusion detection is suggested in this study, which combines dynamic malware pattern analysis with rapid node behavior analysis. In systems that are constantly changing and experiencing novel behaviors, this paradigm enables a significant decrease in false positive alarms. It is possible to accurately detect malware by analyzing dynamic malwares of various types and taking node behaviour into account. Notifications are sent out whenever something that could be harmful happens, so other servers can quickly implement

countermeasures and other systems can be protected from attacks. The self healing system is activated if any malware is detected and the network will be secured without any loss to the network. The proposed model achieves 98.4% accuracy in dynamic malware pattern analysis and self healing model achieves 98.2% accuracy in securing the network from malware attacks and loss of data. In future, hybrid optimization models can be included with deep learning strategies to analyze minute malware patterns also and also to increase the capabilities of self healing systems in large networks by performing the malware pattern analysis and detection in less time.

REFERENCES

[1] Jeon, J., Park, J.H., Jeong, Y.S. (2020). Dynamic analysis for IoT malware detection with convolution neural network model. IEEE Access, 8: 96899-96911. https://doi.org/10.1109/ACCESS.2020.2995887

[2] Roseline, S.A., Geetha, S., Kadry, S., Nam, Y. (2020). Intelligent vision-based malware detection and classification using deep random forest paradigm. IEEE Access, 8: 206303-206324. https://doi.org/10.1109/ACCESS.2020.3036491

[3] Bibi, I., Akhunzada, A., Malik, J., Iqbal, J., Musaddiq, A., Kim, S. (2020). A dynamic DL-driven architecture to combat sophisticated Android malware. IEEE Access, 8: 129600-129612. https://doi.org/10.1109/ACCESS.2020.3009819

[4] Haq, I.U., Khan, T.A., Akhunzada, A. (2021). A dynamic robust DL-based model for android malware detection. IEEE Access, 9: 74510-74521. https://doi.org/10.1109/ACCESS.2021.3079370

[5] Gan, C., Feng, Q., Zhang, X., Zhang, Z., Zhu, Q. (2020). Dynamical propagation model of malware for cloud computing security. IEEE Access, 8: 20325-20333. https://doi.org/10.1109/ACCESS.2020.2968916

[6] Taheri, R., Shojafar, M., Alazab, M., Tafazolli, R. (2020). FED-IIoT: A robust federated malware detection architecture in industrial IoT. IEEE Transactions on Industrial Informatics, 17(12): 8442-8452. https://doi.org/10.1109/TII.2020.3043458

[7] Hong, Q., Chen, H., Sun, J., Wang, C. (2020). Memristive circuit implementation of a self-repairing network based on biological astrocytes in robot application. IEEE Transactions on Neural Networks and Learning Systems, 33(5): 2106-2120. https://doi.org/10.1109/TNNLS.2020.3041624

[8] Liu, Z., Chen, W., Wang, H., Liu, Y. H., Shen, Y., Fu, X. (2017). A self-repairing algorithm with optimal repair path for maintaining motion synchronization of mobile robot network. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50(3): 815-828. https://doi.org/10.1109/TSMC.2017.2726104

[9] Liu, J., Weng, H., Ge, Y., Li, S., Cui, X. (2022). A self-healing routing strategy based on ant colony optimization for vehicular ad hoc networks. IEEE Internet of Things Journal, 9(22): 22695-22708. https://doi.org/10.1109/JIOT.2022.3181857

[10] Akbari-Dibavar, A., Mohammadi-Ivatloo, B., Zare, K., Anvari-Moghaddam, A. (2022). Optimal scheduling of a self-healing building using hybrid stochastic-robust optimization approach. IEEE Transactions on Industry Applications, 58(3): 3217-3226. https://doi.org/10.1109/TIA.2022.3155585.

[11] Waraga, O.A., Bettayeb, M., Nasir, Q., Talib, M.A. (2020). Design and implementation of automated IoT security testbed. Computers & Security, 88: 101648. https://doi.org/10.1016/j.cose.2019.101648

[12] Zhu, H., Li, Y., Li, R., Li, J., You, Z., Song, H. (2020). SEDMDroid: An enhanced stacking ensemble framework for Android malware detection. IEEE Transactions on Network Science and Engineering, 8(2): 984-994. https://doi.org/10.1109/TNSE.2020.2996379

[13] Cai, L., Li, Y., Xiong, Z. (2021). JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. Computers & Security, 100: 102086. https://doi.org/10.1016/j.cose.2020.102086

[14] Ma, H., Tian, J., Qiu, K., Lo, D., Gao, D., Wu, D., Baker, T. (2020). Deep-learning–based app sensitive behavior surveillance for Android powered cyber–physical systems. IEEE Transactions on Industrial Informatics, 17(8): 5840-5850. https://doi.org/10.1109/TII.2020.3038745

[15] Ferrag, M.A., Maglaras, L., Moschoyiannis, S., Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications, 50: 102419. https://doi.org/10.1016/j.jisa.2019.102419

[16] Wang, X., Li, C. (2021). Android malware detection through machine learning on kernel task structures. Neurocomputing, 435: 126-150. https://doi.org/10.1016/j.neucom.2020.12.088

[17] Aonzo, S., Georgiu, G.C., Verderame, L., Merlo, A. (2020). Obfuscapk: An open-source black-box obfuscation tool for Android apps. SoftwareX, 11: 100403. https://doi.org/10.1016/j.softx.2020.100403

[18] Chen, X., Li, C., Wang, D., Wen, S., Zhang, J., Nepal, S., Ren, K. (2019). Android HIV: A study of repackaging malware for evading machine-learning detection. IEEE Transactions on Information Forensics and Security, 15: 987-1001. https://doi.org/10.1109/TIFS.2019.2932228

[19] Liu, Z., Wang, R., Japkowicz, N., Tang, D., Zhang, W., Zhao, J. (2021). Research on unsupervised feature learning for android malware detection based on restricted Boltzmann machines. Future Generation Computer Systems, 120: 91-108. https://doi.org/10.1016/j.future.2021.02.015

[20] Chen, D., Wawrzynski, P., Lv, Z. (2021). Cyber security in smart cities: A review of deep learning-based applications and case studies. Sustainable Cities and Society, 66: 102655. https://doi.org/10.1016/j.scs.2020.102655

[21] Millar, S., McLaughlin, N., Martinez del Rincon, J., Miller, P., Zhao, Z. (2020). Dandroid: A multi-view discriminative adversarial network for obfuscated android malware detection. In Proceedings of the tenth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, pp. 353-364. https://doi.org/10.1145/3374664.3375746

[22] Alzaylaee, M.K., Yerima, S.Y., Sezer, S. (2020). DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, 89: 101663. https://doi.org/10.1016/j.cose.2019.101663

[23] Ren, Z., Wu, H., Ning, Q., Hussain, I., Chen, B. (2020). End-to-end malware detection for android IoT devices

using deep learning. Ad Hoc Networks, 101: 102098. https://doi.org/10.1016/j.adhoc.2020.102098

[24] Sharmeen, S., Huda, S., Abawajy, J., Hassan, M.M. (2020). An adaptive framework against android privilege escalation threats using deep learning and semi-supervised approaches. Applied Soft Computing, 89: 106089. https://doi.org/10.1016/j.asoc.2020.106089

[25] Pektaş, A., Acarman, T. (2020). Deep learning for effective Android malware detection using API call graph embeddings. Soft Computing, 24: 1027-1043. https://doi.org/10.1007/s00500-019-03940-5

[26] Bai, H., Xie, N., Di, X., Ye, Q. (2020). Famd: A fast multifeature android malware detection framework, design, and implementation. IEEE Access, 8: 194729-194740. https://doi.org/10.1109/ACCESS.2020.3033026

[27] Feng, J., Shen, L., Chen, Z., Wang, Y., Li, H. (2020). A two-layer deep learning method for android malware detection using network traffic. IEEE Access, 8: 125786-125796. https://doi.org/10.1109/ACCESS.2020.3008081

[28] Wang, W., Wei, J., Zhang, S., Luo, X. (2019). LSCDroid: Malware detection based on local sensitive API invocation sequences. IEEE Transactions on Reliability, 69(1): 174-187. https://doi.org/10.1109/TR.2019.2927285

[29] Wu, Q., Li, M., Zhu, X., Liu, B. (2020). Mviidroid: A multiple view information integration approach for android malware detection and family identification. IEEE MultiMedia, 27(4): 48-57. https://doi.org/10.1109/MMUL.2020.3022702

[30] Jahromi, A.N., Hashemi, S., Dehghantanha, A., Parizi, R.M., Choo, K.K.R. (2020). An enhanced stacked LSTM method with no random initialization for malware threat hunting in safety and time-critical systems. IEEE Transactions on Emerging Topics in Computational Intelligence, 4(5): 630-640. https://doi.org/10.1109/TETCI.2019.2910243

[31] Feng, R., Chen, S., Xie, X., Meng, G., Lin, S.W., Liu, Y. (2020). A performance-sensitive malware detection system using deep learning on mobile devices. IEEE Transactions on Information Forensics and Security, 16: 1563-1578. https://doi.org/10.1109/TIFS.2020.3025436

[32] Sun, C., Zhang, H., Qin, S., Qin, J., Shi, Y., Wen, Q. (2020). Droidpdf: The obfuscation resilient packer detection framework for android apps. IEEE Access, 8: 167460-167474. https://doi.org/10.1109/ACCESS.2020.3010588