Hindawi
Security and Communication Networks
Volume 2017, Article ID 4749085, 24 pages
https://doi.org/10.1155/2017/4749085

Research Article

A Consensus Framework for Reliability and Mitigation of Zero-Day Attacks in IoT

Vishal Sharma, Kyungroul Lee, Soonhyun Kwon, Jiyoon Kim, Hyungjoon Park, Kangbin Yim, and Sun-Young Lee

Department of Information Security Engineering, Soonchunhyang University, Asan-si 31538, Republic of Korea

Correspondence should be addressed to Sun-Young Lee; sunlee@sch.ac.kr

Received 13 July 2017; Accepted 4 October 2017; Published 13 November 2017

Academic Editor: Antonio Skarmeta

"Internet of Things" (IoT) bridges the communication barrier between the computing entities by forming a network between them. With a common solution for control and management of IoT devices, these networks are prone to all types of computing threats. Such networks may experience threats which are launched by exploitation of vulnerabilities that are left unhandled during the testing phases. These are often termed as "zero-day" vulnerabilities, and their conversion into a network attack is named as "zero-day" attack. These attacks can affect the IoT devices by exploiting the defense perimeter of the network. The existing solutions are capable of detecting such attacks but do not facilitate communication, which affects the performance of the network. In this paper, a consensus framework is proposed for mitigation of zero-day attacks in IoT networks. The proposed approach uses context behavior of IoT devices as a detection mechanism followed by alert message protocol and critical data sharing protocol for reliable communication during attack mitigation. The numerical analysis suggests that the proposed approach can serve the purpose of detection and elimination of zero-day attacks in IoT network without compromising its performance.

1. Introduction

The communication networks are expected to go beyond 40% of the total devices in 2020, which were active in 2012 [1]. All these entities are grouped and studied as "Internet of Things" (IoT). IoT is now a common name and soon it will be a part of daily networking. IoT aims at reducing the gap between the isolated devices and service providers by forming a local network between them.

IoT includes computing components that are grouped together as a single subnetwork on the basis of similar functionalities. These subnetworks have recurrent operations and depend on a common firmware for their activities. A common firmware helps to integrate a vast range of communication devices irrespective of the technology being used by them. Such type of deployment allows easy integration as well as maintenance of IoT devices. A common platform allows easier updates as well as control on the behavior of IoT devices. In general, IoT devices operate in a secure environment and rely much on the security of firmware. However, in certain scenarios, their firmware is subject to various kinds of computing threats, which can infiltrate the operational defense of these devices [2].

The level of threats and possibility of perimeter breaking depend on the types of applications as well as the types of security mechanisms opted by an IoT network. The applications which are dependent on privileged access to a user for its operations are more prone to computing threats. These threats can be known or unknown attacks and depend entirely on IoT activities, such as diagnosis of IoT devices, fault management, firmware updates, mobility management, and information dissemination. Such activities are performed with access to crucial layers of IoT networks, which may provide unauthorized access to notorious groups leading to exploitation of their regular operations [3–5].

All the attacks which are known so far are applicable to IoT networks. The current enterprises are looking forward to nonhuman intervention in network setups. However, with everything being controlled by a computing entity, that too connected via a common network, the risk of falling prey to known or unknown attacks increases [6–9]. Such conditions

demand efficient security solutions for protected operations of these networks.

Most of the business houses and technology developers focus on a common platform that can serve the purpose for "connectivity to all." However, the common platform comes with a workload of management and security issues. Since these operations are only applicable via software-defined solutions, such approaches make the system vulnerable to all kinds of software threats [67]. One of these threats is "zero-day vulnerabilities." These are extremely dangerous vulnerabilities in the network, which if left unidentified may lead to a zero-day attack, that may fail the entire network [52, 68].

Zero-day vulnerabilities exploit the network on the basis of their identification. If the security patches are released after the identification of such vulnerabilities by the notorious groups, there can be serious consequences. On the contrary, identification by in-house administrators can help in mitigation of these threats before making any public announcements. Thus, mode of identification and time of identification are the key role players in the case of zero-day attacks in IoT networks. A software bug can be identified during its testing phase; however, in some cases, it may get unnoticed and gets exploited after a long time of use. Such scenarios are dangerous and convert a zero-day vulnerability into a possible zero-day attack. The name is coined by considering the negligible time offered to the developers or the service providers for counterfeiting these vulnerabilities after their first identification.

IoT networks are highly sensitive as these may possess crucial currency exchange as well as health information of an individual. Thus, zero-day threats are one of the biggest issues for these networks. It is of utmost importance to analyze the operations of these networks and provide an efficient solution for their mitigation. In recent years, most of the researchers have focused on identification of zero-day attacks in IoT networks by analyzing the difference in operations of devices from their normal routine.

The existing approaches, such as detecting advanced persistent threats [64], self-protecting systems [65], and behavior information-based detection [66], are effective, but their scope is limited to detection. These approaches are unable to support communication during threat identification. Also, there are no evident mechanisms for alerting other nodes of the network about the attack formation. Further, these approaches do not account for reliable connectivity, which can continue its operations irrespective of the attack and number of nodes in the network.

In this paper, a study on zero-day attacks is presented followed by a solution, for the issues illustrated in the previous paragraph, which is governed by a consensus framework. The proposed consensus framework relies on behavior context of IoT devices for identification of attacks. The proposed approach not only identifies the zero-day threats in the network but also supports communication and alert messages once an attack is identified. The proposed approach uses two different protocols, one for alert messages and the other for critical data sharing during attacking conditions. The proposed approach is highly scalable and reliable even if the entire network is under the threat of zero-day vulnerabilities. Reliability and consensus cost are the two driving factors of the proposed approach. The results presented for network formulations and latency show that the proposed approach can successfully mitigate the zero-day attacks in IoT network without compromising its performance.

1.1. Background to Zero-Day Attacks. The term "zero-day" is coined considering the negligible time stipulated for overcoming the effects of identified vulnerabilities in a particular software. The number of days for which an anomaly remains unknown affects the countermeasures as well as its effects. This is directly affected by the steps taken to eliminate a known vulnerability. Users with less experience and those who delay the application of patches are the ones who suffer the most. The announcement of vulnerability should be immediately followed by patching of security updates for the identified software. Failing in doing so may lead to harsh consequences of zero-day attacks [69].

The mode of detection has a direct influence on the threat implications of zero-day attacks. Once identified by good guys (white hat hackers), these vulnerabilities can be overpowered with the timely release of security patches; however, identification by bad guys (black hat hackers) makes these systems prey to a different kind of known as well as unknown attacks. Apart from these concerns, the users of a computing entity play a crucial role and can reduce the threat level by following the guidelines for security updates.

Zero-day attacks have a direct influence on the working of an organization and nation. During last decade, a large number of zero-day vulnerabilities were identified in the government of many nations. Most of them were also released on the public domains, which allowed hackers to exploit their parent software by using some known attacks. The announcements of vulnerabilities on the public domains worsen the conditions as all the bugs are easily highlighted which attracts notorious groups of society leading to devastating effects. Such situations also affect the strategies of nations and their mutual agreements.

Every zero-day vulnerability does not lead to zero-day attacks. The ones whose security measures fall short of all possible attacks lose the game and get exploited. These shortcomings of an organization and an individual for implementing security mechanisms give hackers an exploitation window which may lead to zero-day attacks.

The vulnerability cycle for zero-day attacks varies for different situations. In some cases, these attacks are launched covertly once a bug is identified, while, in other cases, hackers may lead to an overt operation by publicizing over different domains [70, 71]. Thus, it is evident that not only do hackers lead to zero-day attacks, but also the delays in updating of security mechanisms also cause possibilities of zero-day attacks. The life cycle of zero-day attacks is studied with "Window of Vulnerability" (WoV). It is evaluated as a software timeline considering the discovery phase, security patching, intermediate exploitation phase, and patch applicability phase, as shown in Figure 1 [52, 70, 72].

The four phases of WoV are split into three sections (operations) as shown with different colors in Figure 1. The

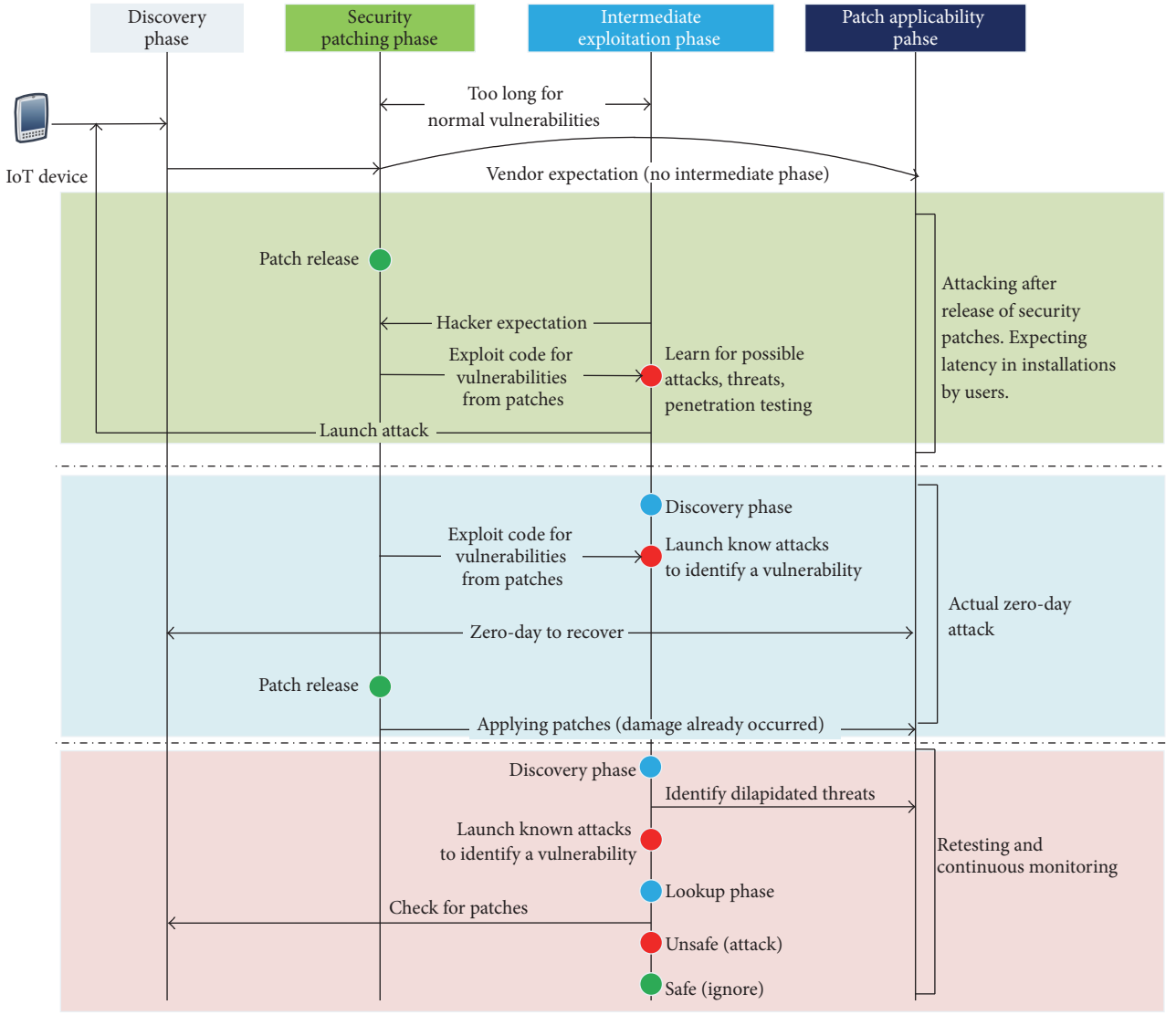


Figure 1: An illustration of the Window of Vulnerability (WoV) for zero-day attacks.

expectation of the vendors is a direct mapping between the security patching phase and its applicability once a vulnerability is discovered. However, such is possible only in the ideal case, which is not applicable for most of the scenarios. The figure shows that, in the first partitioning, after the release of a patch, the hackers expect to identify a bug in the patch by covert or overt operations. This is followed by the launch of known or unknown attacks expecting that users have not updated these patches and there might exist a possibility for zero-day attacks. The second partitioning is when the zero-day vulnerabilities are identified by the hackers themselves and the damage occurs prior to the launch of security patches. The third partitioning is the iterative procedure believing that there might be some vulnerabilities which are left unhandled. These are followed by a decision on safe and unsafe nature of the released patches.

No system is capable of providing a fool-proof security. Exploitation of zero-day vulnerabilities is a matter of time and range of access. There might exist a bug which is left unnoticed for many years, while there might be others that are identified without many interventions. It is of utmost importance for the security analysts, network administrators, penetration testers, and network evaluators to covertly identify such issues and securely share the patches against the identified vulnerabilities.

1.2. Threat Implications and Detection of Zero-Day Attacks. Ever since the computing entities are capable of supporting connectivity between them, the level of threats has arisen to many folds. These threats leave a huge impact on the IoT systems which vary from a simple capturing to destruction and disruption of services [7]. However, these are the worse scenarios; there are other implications associated with the threats in IoT that can act as a silent killer in the form of zero-day attacks. These include the following:

(i) *Ethical Distortion*. Once a zero-day attack is launched, it destroys the usability ethics associated with any

software. Exploitation of a zero-day vulnerability may affect the credibility of an organization leading to huge impact on the economic growth. Apart from destroying and capturing, zero-day attacks raise various ethical issues concerning the usability of a particular IoT device. Zero-day attacks are also used by organizations to destroy the reputation of other organizations leading to a strong ethical conflict and effects on customers and services.

(ii) *Trust Violations and Initiation of Other Attacks*. Zero-day attacks violate the trust and usability rules of any IoT device. Trust violations can cause immediate shutdown of secure services between the implanted IoT devices. These attacks can also release the personal information of an individual or organization on the public domain that makes it vulnerable to different other attacks leading to a major threat of full cyberwar. The network between the IoT devices is assumed to be validated on the basis of a trust as maintained by a particular entity or the service providers. An attack on the firmware or application software can directly affect the security perimeter of an organization. Further, such scenarios can easily influence the remote procedures which are considered to be the backbone of IoT networks.

(iii) *Sovereignty*. IoT devices have seen a tremendous growth in civilian as well as military activities. With a large number of devices involved in surveillance and security services, access to vulnerability can affect the sovereignty of a country leading to serious threats to the public as well as classified domains. During the last few years, the vulnerability in the management software of security services has led to stealing of classified information and its release over the public domain. Such cases are some of the serious implications of zero-day attacks in IoT.

(iv) *End Solutions' Credibility*. Nowadays, most of the enterprises focus on providing a single updatable version of software solutions without requiring reinstallation of firmware. In the case of IoT, such approach can easily be affected if a zero-day vulnerability is identified in endpoint software. The effects can vary and will depend on the actions regarding the updating of firmware. A zero-day attack or exploitation can certainly be harmful to any organization and can destroy its whole chain of software solutions. This is also related to the earlier point on trust violations.

(v) *Defenselessness*. In some cases of zero-day attacks, cascade vulnerabilities are identified leading to multiple zero-day threats. Such conditions in IoT networks can make the entire perimeter defenseless despite the strengths of firewall and other detection solutions. Such scenarios require large-scale alterations in the entire network and may require network replanning leading to huge cost overheads on the organizations.

(vi) *Conceptual Disfigurement*. The success of IoT networks lies in the depth of concept and planning. With zero-day vulnerabilities, the concept of security gets destroyed leading to the disfigurement of the entire idea. Such conditions also expose the strategic context as well as instances involved in decision-making leading to deformity at personal as well as team levels. Further, such attacks vandalize the IoT device in their deployment network without any use.

As discussed earlier, zero-day attacks do not give enough time to security analyzers and developers for overcoming the threats. It is difficult to identify and trace such threats even after years of development. However, covert analyses on the periodic basis can prevent the exploitation of zero-day vulnerabilities. Further, reverse engineering has also increased the chances of occurrence of such attacks. Detection of zero-day attacks in IoT should be fast and it should provide less time to hackers for developing exploit codes.

Identification of unknown attacks and behavior variations of each IoT device make it difficult to provide any countermeasures against such threats. Embedded IoT devices need to be secured by the means of secure embedded coding solutions [5]. Signature analysis and code validation should be performed periodically to analyze the frequency of a zero-day vulnerability in IoT systems. Evaluation of RFID security protocol and packet analyses should also be carried out for detecting any vulnerability. The remote procedures associated with the IoT updating process should be protected using channel security. However, this cannot prevent the zero-day attacks entirely but can ensure a latency to attackers in gathering information from the remote procedures [72]. Further, various techniques available for detecting zero-day possibilities in a network are listed in Table 1. The table discusses different types of detection approaches with their motivation, description, and applicability issues. The existing approaches can be classified into pattern-based detection, heuristic-based detection, reputation-based detection, behavior-based detection, virtualization-based detection, and irregular symptom-based detection. Most of these approaches rely on finding vulnerabilities in the exploited system but are not well-matched for real-time detection. Further, the communication between the devices after the identification of an attack is not discussed much in these solutions.

1.3. Motivation and Our Contribution. The devices in IoT networks are highly sensitive and depend on a common platform for the majority of their operations. With a common firmware from their business operators, these devices are operable on software technologies, which may fall prey to some bugs. These bugs may be noticed after a long period of time, or, as in some cases, these are ignored during testing phases. Such scenarios lead to vulnerabilities that can be exploited by the black hats for harming the operations of IoT networks. Thus, prevention of network against the known or unknown attacks which can be launched at any time in the near future is the motivation behind the development of the proposed framework. There are a lot of approaches for identifying zero-day attacks with a high accuracy, but (to the best of our knowledge) most of these are not able to withstand the communication burden and may halt the network during

Table 1: Zero-day detection techniques applicable in IoT.

Detection techniques	Motivation or background	Description	Issues
Pattern-based [10–17]	To detect and analyze malicious codes incoming promptly from outside	After defining the specific pattern of existing malicious codes as their characteristics, malicious codes are detected and blocked by matching defined pattern with a pattern of incoming codes	(i) This technique can support fast detection by just comparing defined signatures (ii) However, new and variant malicious codes are not defined in pattern and are not detected
Heuristic-based [18–24]	To detect and analyze new and variant malicious codes	This technique determines specific behavior of malicious codes, so this can check new and variant codes by analyzing abnormal behavior not signatures	(i) It is hard to define criteria for comparison of similarities of abnormal behavior (ii) This causes false positive by detecting normal program as malicious code
Reputation-based [25–28]	To detect and analyze new and variant malicious codes	(i) This technique is similar to pattern-based technique (ii) In particular, if new malicious codes are emerged, reliability is determined based on feedback for opinions of a large number of users (iii) Reputation information is defined on the basis of number of users, manufacturer of codes, etc. (iv) Accuracy and reliability depend on possession and analysis of a large amount of reputation information	(i) Accuracy and reliability are only dependent on user's opinions (ii) If reputation information is not enough, accuracy and reliability are decreased
Behavior-based [29–42]	(i) Signature, pattern, and reputation information are hard to analyze by malicious code analyst preferentially (ii) There is a limitation to collect or analyze malicious codes when the number of codes increases exponentially (iii) It is a difficult approach for analyzing malicious codes realistically because of reasons that the rate of malicious code generation is much faster than the speed of analysis	(i) This technique detects faulty behavior when malicious codes are executed (ii) This is an improved version of heuristic-based technique (iii) Malicious behavior is revealed not only in executable files, but also in document files, such as PDF, DOC, and HWP (iv) This technique determines characteristics of malicious behavior based on file, registry, network, process, etc.	The system can be infected because behavior is analyzed during execution process in the actual system
Virtualization-based [43–48]	An environment to analyze malicious behavior in a separated space from actual system is required	(i) This approach is closely related to dynamic heuristic-based technique (ii) Malicious codes are analyzed in virtual system	This does not detect attacks efficiently and takes a lot of time to penetrate the system, even after collecting various pieces of information and utilizing unknown attacks
Abnormal/irregular symptom-based [49–51]	To detect and analyze unknown and zero-day attacks	(i) To detect abnormal behavior, this technique collects and integrates logs, which are generated in the system, and analyzes the correlated information (ii) In particular, this is required for detecting infected systems, with unknown new malicious code, and it helps to determine whether transmitted traffic is normal or abnormal	(i) In case of security system, if the system does not process information in real time, the system gets exposed to security threats (ii) It is technically difficult to collect and analyze correlation for high-capacity and high-speed traffic from network

attack scenarios. This also motivated the need for a common solution which can handle both the identification of zero-day attacks as well as reliable communication during such possibilities.

The proposed approach forms a consensus framework which uses behavior context of IoT devices for identifying zero-day vulnerabilities in the network. Next, the proposed approach uses alert message protocol for notifying network components about the infected nodes as well as subnetworks. Finally, the proposed approach uses a critical data sharing protocol for disseminating information despite the presence of zero-day vulnerabilities. The proposed approach uses reliability and consensus cost to form their solution and it also suggests mechanisms for security patching and reestablishment of trust.

The rest of the paper is structured as follows: Section 2 presents related works. Section 3 gives insight into the network model, proposed approach, and protocols. Section 4 presents the performance analysis. Section 5 concludes the paper with a future scope.

2. Related Works

Zero-day attacks have been studied by a lot of researchers in different forms and environments. The existing works have focused on the elimination of these attacks on the basis of fixed parameters, such as false positive ratio, false negative ratio, accuracy, and the area under curves. These approaches operate towards the elimination of possible zero-day threats in a connected environment. However, very few of them have highlighted the consequences and solutions for zero-day in IoT networks. Intelligent solutions are required for countering these threats and preserve communication despite the level of damage. Machine learning can be used as one possible ground for intelligent mechanisms that can help to understand threat implications and provide a suitable remedy accordingly [73].

Recently, Singh et al. [66] proposed a solution for detecting zero-day attacks by using device signatures and behavior. The approach uses a hybrid technique for detecting threats. This solution can be useful in analyzing streaming data against zero-day threats but is unable to provide a reliable communication mechanism for rest of the network. Duessel et al. [74] focused on application-layer based zero-day attack detection. The authors relied on investigating anomalies by using contextual information. The authors used text-based solutions and binary application-layer protocols for detecting zero-day anomalies. Similar to other existing techniques, this approach is limited to detection over a node and does not account for network attacks and management when encountered with zero-day vulnerabilities.

Chamotra et al. [75] suggested that honeypot baselining can be an efficient solution for analysis in an attacker environment. The authors particularly emphasized the use of honeypot baselining in detecting zero-day attacks. Their approach is based on the XML mapping, but this can be manipulated if the structure of files is known to the attackers. How the rest of the network will operate is still an open issue with this solution. Sun et al. [76] proposed a probabilistic solution for zero-day attacks. Their approach is based on high infection probabilities for detecting zero-day paths. With a proper network-based extension, this can be a useful solution in mitigating extreme threat implications of zero-day attacks. However, with an erroneous measurement of probability, this approach can lead to excessive computations, which is not desirable.

Self-protecting computing systems, as suggested by Chen et al. [65], are desirable for detecting zero-day threats in an IoT environment. The authors gave a prototype by using virtual machines for building self-protecting systems. Their solution is efficient and can be considered for extending to IoT networks. However, the present state of this approach requires a much contextual evaluation of devices before considering it for zero-day analysis of IoT networks. Host-based intrusion detection system can be used for detecting zero-day attacks [77]. Such solutions use a window-based dataset to form a system capable of detecting zero-day threats by analyzing normal as well as abnormal data. This kind of solutions has limited scope and can be manipulated by multiple attacks. Also, such solutions need a considerable extension for real-time analysis of zero-day attacks in IoT networks.

Apart from the related works presented above, some key aspects of general zero-day attacks and zero-day attacks in IoT are presented in Tables 2 and 3, respectively. Most of the existing solutions, discussed in these tables, focus on identifying programming errors by using strong debugging, executable operations through attachments, data forging through authentication failures, and so on. Further, the subsisting solutions are vulnerably susceptible to threats as the analysis phase itself can lead the system to zero-day attacks. Apart from these, the dependence on data accumulating and lateral analysis does not ensure a real-time solution for zero-day susceptibility detection in IoT networks.

From the study presented in this paper, it is evident that existing solutions are capable of identifying zero-day attacks by using various mechanisms, but (to the best of our knowledge) most of them are unable to provide a strategy for reliable communication during these attacks. Also, these approaches lack aftermath of zero-day exploitation.

3. Proposed Work

The proposed approach aims at mitigating aftereffects of zero-day attacks and providing a strategy for analyzing the communication across the entire network. With the efficient identification of falsifying groups, the proposed approach uses a consensus framework for reliable communication even during zero-day attacks.

3.1. System Model. The network comprises a set $\mathscr{M}$ of IoT devices out of which some are independent while others are Corresponding Nodes (CNs). The CNs operate as a single network under a control of common Home Gateway (HGW) and a network can have multiple HGWs represented by a set $\mathscr{H}$. The devices under each HGW are represented by $M_1, M_2, \ldots, M_{|\mathscr{M}|}$, such that $\mathscr{M} = \bigcup_{i=1}^{|H|} M_i$. Further, the

Table 2: Zero-day attacks.

Attack techniques	Mechanism	Focus	Methodology
Network attack vector [52, 53]	Launch the attack's malicious payload and propagate itself	Programming errors	Protocols and network-aware processes
Application attack vector [52, 53]	Launch executable files	Open e-mail attachment	Executable files
Control system attack vector [54]	Destroy control system such as SCADA and PLC	Server Service (MS08-067), Windows Shell (MS10-046), Print Spooler Service (MS10-061), and Windows Kernel-Mode Drivers (MS10-073)	Third parties, LAN, and removable flash drives
Worm propagation [54]	Propagate worms or bots inside the network	Infection of Web server	IIS
Targeted attack (APT) [54]	Penetrate targeted system	Misplaced diversity	Weakest path
Moving target [54, 55]	Evade antivirus detection	Limit the exposure of attackers and opportunities and mitigate system resiliency	Continually shift and change over time to increase complexity and cost

Table 3: Zero-day attacks in IoT.

Attack techniques	Mechanism	Focus	Methodology
Asynchronous attack [56]	Lead to instability of a real-time energy market	Erroneous control	Desynchronizing of smart meters
Simple packet delay attack [56]	Desynchronizing the slave nodes	Manipulate of a slaves' clock	Delay of the transmissions of the NTP or PTP packets
DDoS [57, 58]	Denial of services, service unavailable	Unknown or new attack, exploiting vulnerabilities, overload resources	UDP flood, ICMP/PING flood, SYN flood, Ping of Death, etc.
Advanced persistent threats (APT) [58]	Unauthorized person attempts to gain access to the system	Stealing data	Bypassing authentication
Man-in-the-Middle (MITM) attack [58–60]	Gain illegitimate access to the system or the network	A program or person masquerades as another program or person	Spoofing and sniffing attack
Replay attack [58, 61, 62]	Disguise valid entities or messages	Bypassing integrity	Valid message containing some valid data is repeated again and again

network uses a set $\mathscr{B}$ for Base Stations (BSs) and a set $\mathscr{A}$ for Access Points (APs) for communicating with IoT device directly or indirectly via HGW. The network includes a set $\mathscr{N}$ of Mobile Nodes (MNs) which aim at sharing context as well as information with IoT devices. The network also has inter-IoT device communication between all the elements of set $\mathscr{M}$. An illustration of the initial and proposed network architectures is shown in Figure 2 with hierarchical view in Figure 3.

The problem in this paper is presented as cost-reliability formulation [78]. The cost involves the "cost of consensus," whereas the reliability is the belief a node exhibits on every other entity in the network including itself. Both of these parameters are used to derive the level of connectivity, which helps to analyze whether a connection exists between any two nodes as well as helping in predicting the possibilities of connectivity on the basis of given behavior of a node. The proposed system is affected by the number of diagnosis systems that monitor a single node. In the network, considering that every infrastructure possesses a diagnosis system, the total number of subnetworks γ will be equal to $|\mathscr{H}| + |\mathscr{A}'|$, where $\mathscr{A}'$ is the set of APs that do not interact with any HGW. With α being the initial connectivity constant between any two entities, $\mathscr{I}$ being the set of diagnosis system in charge for a given set of nodes, s being the alternative routes for each device, and τ being the trustworthy components, the reliability $\mathscr{R}$ of the network can be calculated as [78]

$$\mathscr{R} = \prod_{i=1}^{\gamma}\left(1 - \sum_{j=1}^{|\mathscr{I}|}\sum_{\tau}\prod_{k=1}^{s_i}\left(\binom{\mathscr{E}_{i,k}}{\tau'}\alpha_{ik}^{\tau'}\left(1-\alpha_{ik}^{\mathscr{E}_{ik}-\tau'}\right)\right)\right). \quad (1)$$

Here, τ' is the difference between the actual connections and the active connections available for a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ formed between the network entities, such that $\mathscr{V}$ is the

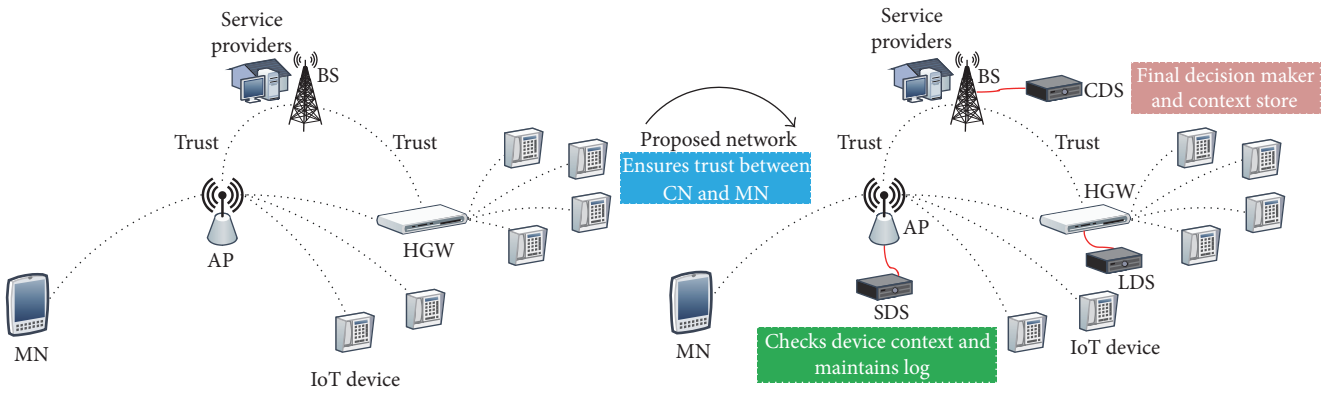


Figure 2: An illustration of the initial and proposed network architectures.

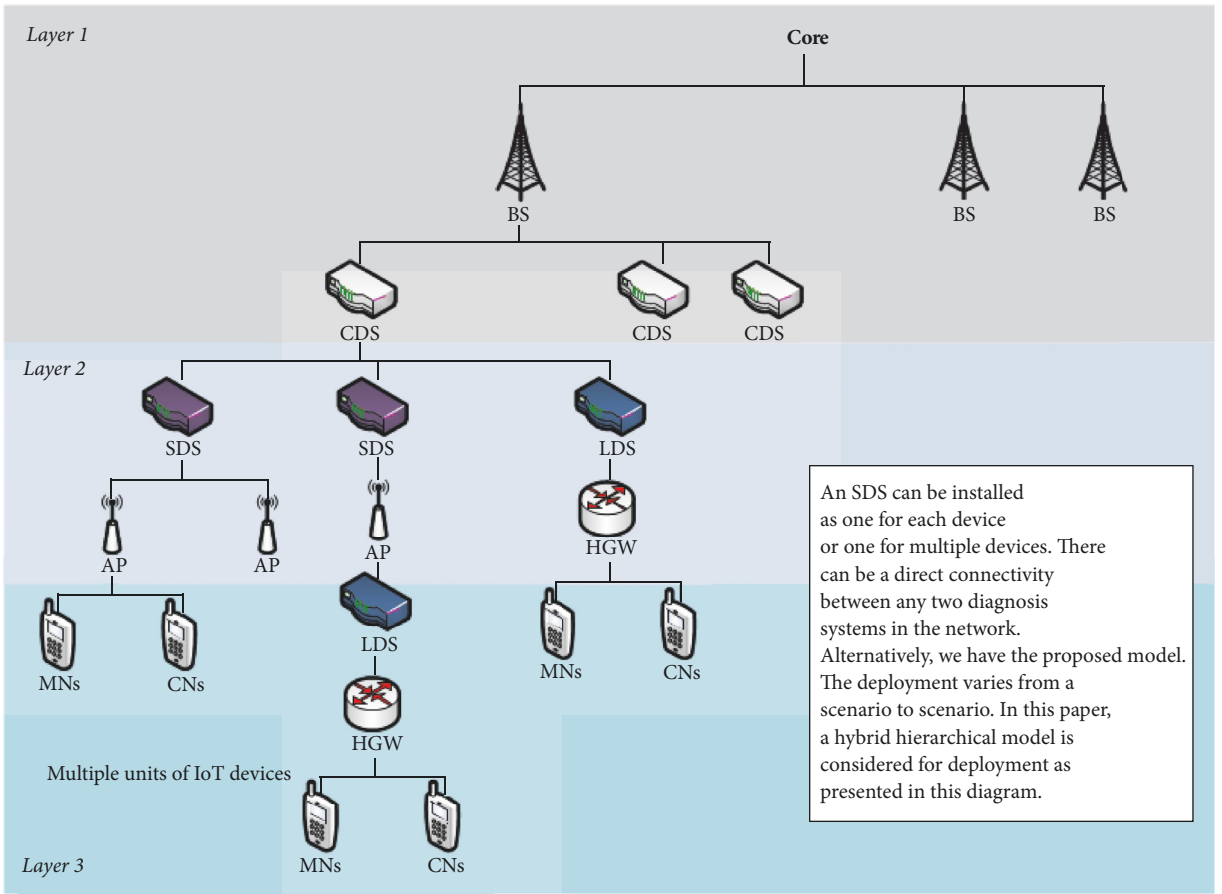


Figure 3: An illustration of hierarchical view of the defined network.

union of all the network entities and $\mathscr{E}$ refers to the set of connections per node. Let $\mathscr{W}$ be the weight associated with every link which is active in the network, such that, for a single instance, $\max(\mathscr{W}) = |\mathscr{E}|((|\mathscr{E}| - 1)/2)$. In the proposed approach, the consensus cost means the minimum cost of connectivity that enables maximum reliability for maintaining communication in the network. The consensus cost is derived as a mutual cooperation between $\mathscr{E}$, γ, halt time $\mathscr{T}$, and τ'. In the proposed approach, the mitigation of zero-day attacks is followed by realization of reliable communication between the nodes. Thus, for optimal solution, the proposed approach relies on following conditions:

$$\begin{aligned} \mathscr{R} &= \max, \quad C_o = \min, \\ \mathscr{R} &= \mathscr{R}_{\mathrm{TH}}, \quad C_o = \min, \end{aligned} \tag{2}$$

where $\mathscr{R}_{\mathrm{TH}}$ are the threshold limits for the reliability of the network during attacks and C_o is the consensus cost defined

over multiple parameters, such that, for given instances η,

$$C_o = \mathscr{W}\mathscr{T} \times \left(1 - \frac{\tau'}{\max(\mathscr{E})}\right)^{\eta}, \quad (3)$$

provided that all the variables are at minimum even after a vulnerability is exploited.

3.2. IoT Diagnosis System. The proposed approach forms a consensus framework for mitigating the effects of zero-day attacks and provides reliable communication once an attack is identified. In the proposed approach, MNs are secured via trust, which is maintained by the service providers who are also in charge of the BSs. The diagnosis system forms the backbone of the proposed consensus framework and helps in diagnosing the issues at different operational layers of the network. The diagnosis systems are just like the monitoring devices which manage the context of IoT devices and help in maintaining reliability between the network entities. The diagnosis system can be installed centrally or distributively. The details of diagnosis systems are provided below:

(i) *Centralized Diagnosis System (CDS)*. CDS is the main diagnosis system which is installed on the central BS that manages multiple APs as well as HGWs. CDS is the key entity in managing the trust as well as generating updates for the entire network. The updates are shared via security mechanisms and communication protocol provided in the later part of this paper. In general cases, the behavioral aspects of IoT devices are available with the CDS and it shares them with the other diagnosis system whenever required or demanded. CDS is highly active in reliable communication between IoT as well as in alarming devices once a particular attack is registered in its database.

(ii) *Local Diagnosis System (LDS)*. LDS is the near-user diagnosis system which is installed on the HGW. LDS manages the operability of IoT devices which indirectly interact with the BSs via HGW. LDS is critical for some of the applications, such as smart home management system and smart monitoring. LDS also obtains the context of its devices and stores them locally. However, an issue with the LDS may expose the entire network to different kind of attacks. Thus, the information of devices in HGW needs to be encrypted. Also, the details can be stored distributively, by storing ID at HGW and the corresponding information at the CDS. It is assumed that LDS are secured prior to communication and the service providers have sufficient information regarding the operations of LDS.

(iii) *Semi-Diagnosis System (SDS)*. SDS is installed on the AP for supporting the IoT devices which do not act as a CN. SDS interacts directly with the CDS, and, in some cases, it is capable of retrieving information from the LDS. SDS helps in sustaining communication when LDS is irresponsive or it violates the rules of communication. In highly dense networks, SDS is not believed to handle context of devices, and the data collected from them is shared directly with the CDS. Further, the context and data sharing protocols help in checking the authenticity of each device once a vulnerability is exploited in the network.

The proposed approach deals with both scenarios of storing context at the centralized and distributed diagnosis systems. The evaluation part of this paper uses both of these strategies for analyzing the performance of the proposed approach. Also, the level of abstraction and distribution of information depend on the context and amount of data to be shared and transmitted between the IoT devices as well as the network infrastructure.

3.3. IoT Context and Behavior Formulation. The context is a driving force behind the formation of a reliable network which can operate effectively even in the presence of zero-day attack without exposing the entire network. The context helps in forming a strategy which can be employed for generating trust rules between the IoT devices. The context may vary for different IoT devices. In this paper, general IoT context is considered in the formation of behavior rules. These rules can be further exaggerated depending on the requirements as well as the application scenario.

The context is decided by the CDS after registering all the devices during initialization of the network. These contexts are stored in every diagnosis system and shared mutually between the demanding diagnosis systems. The values are stored periodically and appended in a single file for every device. The size of storage depends on the level of feedback expected from the network operators as well as on the memory of diagnosis systems. Note that CDS does not possess a constraint to memory, but HGWs and APs have limited memory; thus, logs should be shared with the CDS on a timely basis. In the proposed approach, it is assumed that the CDS, SDS, and LDS are connected via a secure path; thus, security of logs is not a concern in the proposed approach. At present, the general information available for each IoT device is selected as a context. This can be altered depending on the types of devices and network configurations. The details of the context used in the proposed approach are provided below:

(i) *Device Signatures* ($\mathscr{D}_s$). This is the unique ID of a device allocated once it registers itself with the CDS. Note that $\mathscr{D}_s$ are allocated in lieu of physical address of every requesting device. These signatures are embedded in the device and also shared with the corresponding SDS or LDS.

(ii) *Pseudo Signatures* ($\mathscr{S}_s$). This is the pseudo ID generated from the local entity for communication with the fellow devices. It is to be noted that every CN in the network recognizes another device by its $\mathscr{S}_s$. In the adverse cases, the local entity can shuffle these IDs, but with updates to every CN; otherwise, it may lead to sharing of wrong information to a wrong device.

(iii) *Update Counter* ($\mathcal{U}_c$). This is the firmware update counter which tells the number of times the firmware of a device is accessed and updated. The value of $\mathcal{U}_c$ may be different for each subgroup. It is fixed by the CDS by selecting a random value for every requesting subgroup. These counters do not interfere with the performance of the network and help in determining the level as well as the depth of updates performed in the IoT network. The update counters are stored at the central level and local level as well as at the device.

(iv) *Traffic Type* ($\mathcal{F}_t$). This is one of the crucial parameters which helps in deciding the content shared between the IoT devices. $\mathcal{F}_t$ governs the rules of traffic, which is expected from a particular IoT device and focuses on the exact outcomes as intended in the network. Usually, one set of IoT devices operating in the same domain has similar values for $\mathcal{F}_t$. However, depending on the level of interactions, the log for some devices may have multiple entries for $\mathcal{F}_t$.

(v) *Header Length* ($\mathcal{H}_l$). This includes the metadata of the IoT device and its controlling diagnosis system. Header fields are also subjected to some random integers generated by their corresponding diagnosis system. These random integers are selected by the diagnosis systems during the initialization of the network and are periodically updated. For the complex security of network parameter, corresponding diagnosis systems of each device may induce alterations in these random values to make IoT device unaware of the exact settings.

(vi) *Memory Range* ($\mathcal{Y}_r$). This controls the size of packets which are shared between the network entities. $\mathcal{Y}_r$ includes overall size of the packets including the code and binaries. In the case of exploitation of binaries, this metric can help in determining its correctness.

(vii) *Route* ($\mathcal{O}_r$). This field helps to check the route followed by the traffic coming from a particular IoT device. The diagnosis system matches the route with the path defined by it. This does not help in determining vulnerabilities but rather helps in checking the path which is followed by a device identified as a potential threat of zero-day attack.

The above context helps in defining the behavior formulations as well as the rules for analyzing the operations of any IoT device subjected to the possibilities of zero-day vulnerabilities. The context above can have fixed or variable lengths and has a lot to do with the content of applicability of a device in practical conditions. The behavior formulation for each device is performed by forming a network of context described above and storing the general values into a tabular log for easier assessment. Various rules that are to be followed for ensuring consensus and trust between the network entities, as well as the diagnosis systems, are as follows:

(i) CDS is the one in charge and has the capacity of altering the context requirements in the network.

(ii) CDS can ensure trusted parties and such entities can be ignored from context matching. However, in the case of critical situations, all the entities fall under the proposed threat elimination categories and must ensure their trust before proceeding with the data exchanges.

(iii) Each diagnosis system has the capacity of deriving its dominance chart for the given set of context and can alter it periodically. This creates an ambiguous situation for the in-house attacker as it is always unaware of the checking rules.

(iv) One-to-one or single context mapping can be done depending on the situations and also on the transmission rates. Since mapping and matching of entire context consume time, unnecessary evaluations may lead to network overheads. However, the proposed approach takes care of such situations and the context strategy is a lightweight that does not cause many overheads.

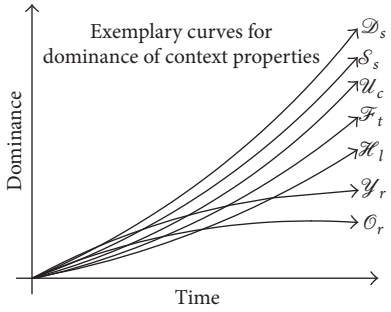
(v) SDS and LDS can operate in dual mode seeking context from the CDS as well as the MN and the CN. They can form a local context decision system on the basis of their diagnosis results for every connection. This provides an extra layer of security but at the cost of excessive computations.

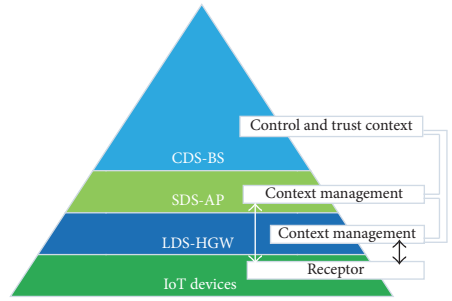
(vi) Any entity whose application access is altered as per its initial configurations needs to register its new format with the CDS as well as the corresponding LDS and SDS.

(vii) LDS and SDS have the capabilities of limiting the access to any device in the network, connected to their subgroups, in the case of alert messages.

The dominance of each context may vary with time and is fixed by the CDS. The dominance helps in retaining control over the evaluations in the network. This allows management of network without burdening the diagnosis systems. An illustration of exemplary curves for context, interaction module, and the context storing table is shown in Figure 4. The curves can follow any trend and the interaction module remains the same. The log and context storing table changes in accordance with the values set by CDS. The contexts from different entities are matched to analyze the behavior of IoT device allowing a platform for the formation of context graphs and strategic decision system.

3.4. Context Graphs and Strategic Decision System. The context graphs and strategic decision system are an integral part of the proposed approach. Both of these account for correct working as well as mitigation of zero-day attacks in IoT network by following the previously defined context rules. Let $\mathcal{G}' = (V', E')$ be the context graph formed for each node, where V' is the number of computational stages including input and output states. The context graphs are derived for every IoT device and the results are stored only for the first (input) as well as the final (output) stages. This helps in saving memory as well as knowing the exact state of every device





In order of dominance

Properties	CDS	SDS*	LDS	IoT	Match
$\mathscr{D}_s$	—	—	—	—	Yes/no
$\mathscr{S}_s$	—	—	—	—	Yes/no
$\mathscr{U}_c$	—	—	—	—	Yes/no
$\mathscr{F}_t$	—	—	—	—	Yes/no
$\mathscr{H}_l$	—	—	—	—	Yes/no
$\mathscr{Y}_r$	—	—	—	—	Yes/no
$\mathscr{O}_r$	—	—	—	—	Yes/no

*NA: if not applicable as per the hierarchy model.

Figure 4: An illustration of exemplary curves for context, interaction module, and the context storing table.

in the network. In some cases, the intermediate stages can be saved. Such mechanisms allow in-depth evaluations by analyzing the working of every IoT device. However, these are subject to memory constraints and may cause many overheads. The context is derived individually depending on the dominance of context properties. In the context graphs, E' refers to the connectivity between the IoT referring to cohesion and coupling between the stages. Low values for context edges mean string-like one-to-one mapping, which is easier to follow and analyze. The size of $\mathscr{G}'$ may vary depending on the complex behavior of an IoT device.

The proposed strategic context graph-based solutions are applicable in the network during the deployment phase, rather than the development phase. However, these must be provided as an inbuilt facility in all the IoT devices. Thus, the proposed strategy comes handy only when a vulnerability is identified in the lateral stages of operations by either the development team or the security analysts. It can also be considered while releasing security patches and updates for a possible zero-day attack in the network. Operating on the behavior of every device, the proposed approach helps to track their stage-wise operations.

The management of IoT devices against the zero-day attacks is much affected by the strategic decisions. For this, CDS follows a rule of updating $\mathscr{U}_c$ with a random value anytime during transmissions. Once the values are decided, these can be used to identify the number of times a firmware of a device has been updated in real time and its actual value has been stored in the device. This helps in understanding if the firmware of a device is manipulated or not. The success of this depends on the exact matching of the context between the device and the diagnosis systems. An illustration of the operational view with an exemplary scenario for context graphs is shown in Figure 5. Once the context is shared between the diagnosis system, a simple rule of mutual exclusion is followed for analyzing values of selected or all contexts. The input and output stages are crucial as these include the entry and exit of intruders in the IoT firmware as well as the subnetwork. More intermediate states for every device allows much time for analyzing the context and behavior of a device. However, if the intermediate states are less or fast, delays can be added in the proposed strategy. The procedures in the proposed approach are described as four stages, as follows:

(i) *Stage 1.* The IoT device interacts with network entities for its registration and permissions for context sharing. The interaction procedures are secured via trust rules and security mechanisms for communications. This is the input stage and is followed by self-processing of IoT devices.

(ii) *Stages 2 and 3.* These are the intermediate stages and are referred to as a self-processing mechanism for the IoT devices. A procedure can have n number of stages depending on the type of context and configuration of the network. Stages 2 and 3 are operated as processing stages between the input and the output stages. These stages are accompanied by information gathering and processing by respective diagnosis systems. A high number of intermediate stages ensures more time for information processing. However, an excessively larger value for n may induce excessive overheads.

(iii) *Stage 4.* This is the final step for taking a decision on any IoT device and invoking a consensus data sharing protocol if intended because of the irregular behavior of IoT device or identification of misleading context. Any misleading device is identified after completion of stage 4 in the proposed approach.

The steps for strategic decisions on the basis of retrieved context through these stages are presented in Algorithm 1. The algorithm monitors the network continuously and its operational complexity is dependent on the amount of context as well as the connected components. Once a possibility of an attack is identified, the proposed approach uses consensus data sharing protocols which operate for alerting the network entities as well as reliable data sharing without being exploited by the zero-day vulnerabilities. The network operating with standard context matches the exactness with constant time, but, in the general case (proposed scenario), the complexity of Algorithm 1 will be given on the count of number of IoT devices and is equal to the complexity of forming a graph $\mathscr{G}'$.

3.5. Consensus Data Sharing Protocols. The consensus protocols are used once a zero-day vulnerability is identified in the network. These protocols can be used in case any node violates the reliability rules of the network. In the proposed approach, two different mechanisms are used for

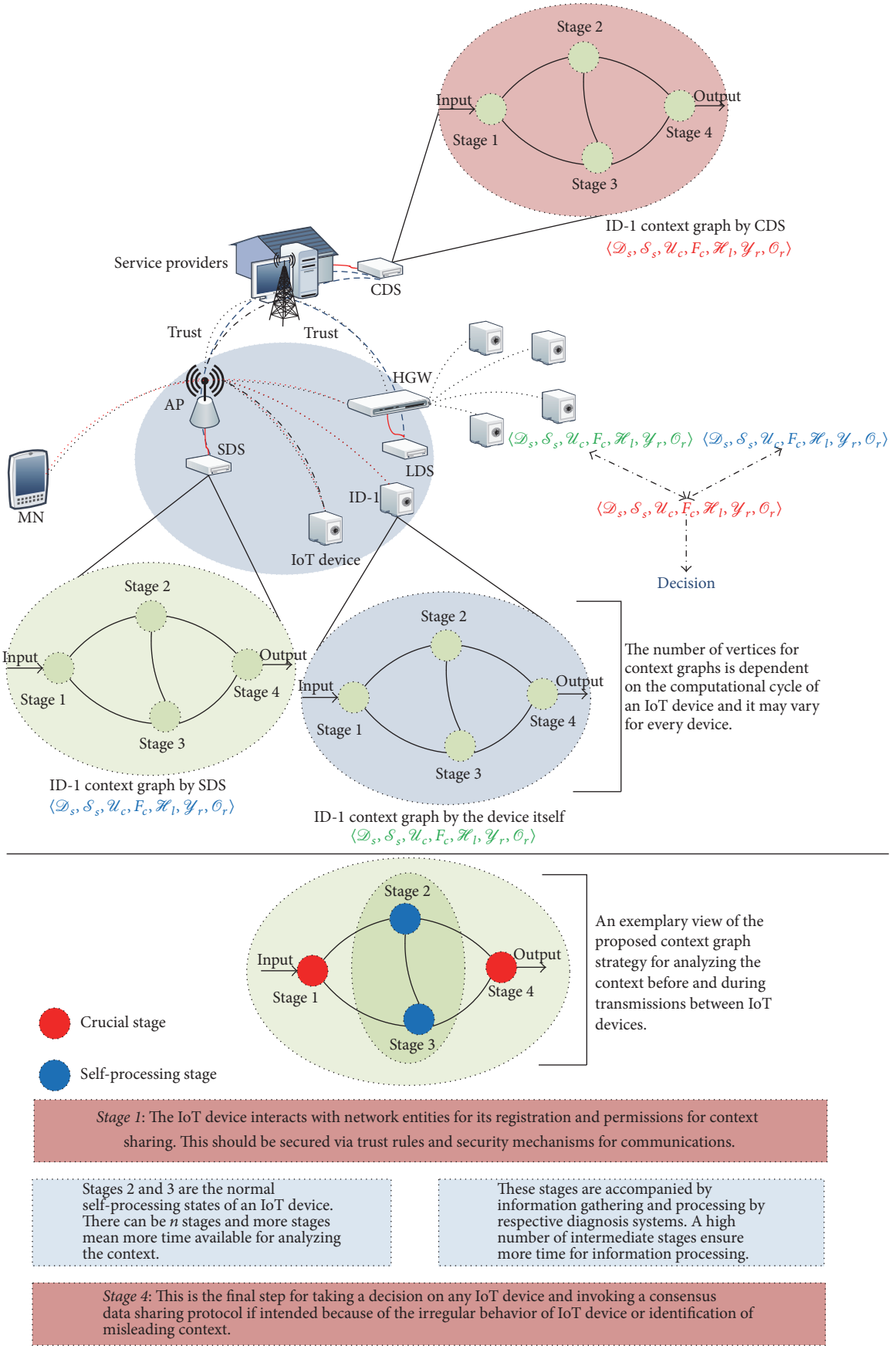


Figure 5: An illustration of exemplary scenario for context graph formation and evaluation of communicating IoT devices.

(1) **Input**: $\mathcal{G} = (\mathcal{V}), \mathcal{E}, \mathcal{D}_s, \mathcal{S}_s, \mathcal{U}_c, \mathcal{F}_t, \mathcal{H}_l, \mathcal{Y}_r, \mathcal{O}_r$
(2) **Output**: context decision
(3) Register IoT devices with CDS, SDS, LDS
(4) set $\mathcal{U}_c$
(5) **while** *transmission*! = *NULL* **do**
(6) set a periodic counter for evaluation
(7) Retrieve context from IoT device
(8) Form $\mathcal{G}' = (V', E')$
(9) Mark input and outputs
(10) Initialize context table
(11) **if** $|\mathcal{V}|$ = min && $\mathcal{R} < \mathcal{R}_{\text{TH}}$ && C_o = max **then**
(12) Mark the device
(13) Check $\mathcal{U}_c$ with CDS, SDS, LDS
(14) Analyze $\mathcal{D}_s, \mathcal{S}_s, \mathcal{U}_c, \mathcal{F}_t, \mathcal{H}_l, \mathcal{Y}_r, \mathcal{O}_r$
(15) Update context table
(16) Generate alert messages (context table)
(17) Follow consensus data sharing protocol
(18) **else**
(19) Continue
(20) **end if**
(21) Operate till trust is not ensured
(22) **end while**
(23) exit

Algorithm 1: Strategic decision on retrieved context.

protecting the network against unaware circumstances as well as intrusions. The details of both the protocols are provided in the following subsections.

(1) Alert Protocol. The alert protocol is invoked as a part of Algorithm 1. This protocol helps disseminate the alert messages across the network, allowing the formation of a secure workgroup. An illustration of the alert messages protocol is shown in Figure 6. The protocol initiates with reports and moves on taking a decision on the behavior of an IoT device. The steps followed by this protocol are explained below:

(i) The first step is the reporting by an MN about the unresponsive behavior of an IoT device to the CDS. The CDS begins investigation procedures which aim at retrieval of information from the required nodes.

(ii) The investigations are followed by the request messages for different context to the SDS and the LDS. The context messages are distinguished on the basis of the difference in the total information required as well as the IDs used for reporting.

(iii) The LDS operates towards fetching of results and logs from every connected device, which responds by sharing their reports.

(iv) Following this, $\mathcal{U}_c$ is checked at the CDS which is received from the LDS and the SDS.

(v) Next, all the diagnosis systems analyze the routes and maintain a report, which is followed by identification of false nodes.

(vi) Finally, a decision is taken and, in the case of unsafe nodes, alert messages are sent to the corresponding diagnosis systems. Using this, LDS selects the problematic device and alerts all other devices about its state.

(2) Critical Data Sharing Protocol. The alert message protocol is followed by a critical data sharing protocol (Figure 7). This protocol helps to disseminate information and maintains data flow even during unfavorable conditions. This protocol operates on an assumption that a secure path exists between all the diagnosis systems. The detailed procedures for this protocol are explained below:

(i) The first step begins with gathering alarming information, which is done by the previous alert protocol. LDS continuously fetches and maintains the log for every connected device.

(ii) All the connected components, namely, SDS, LDS, and MN, send their alarming updates and reports to CDS. This is done to acquire information of the subnetwork as a single CDS may be managing one or more subnetworks.

(iii) Next, the CDS fetches route logs and context from LDS as well as for the intended IoT device from the MN. Once the initial steps are performed, the CDS changes the pseudo IDs of every IoT device and reallocates them to the appropriate device via SDS and LDS.

(iv) The LDS maintains a policy of nondisclosure of new IDs to unsafe nodes. It is to be noted that the unsafe nodes always possess the same ID throughout the transmission that was allocated to them during initialization of the network.

(v) Once done with these steps, the LDS sends failed nodes information across the network and receives log reports and acknowledgments from every connected device.

(vi) Finally, the CDS shares the available information to all the intended entities and allows communication to begin in normal flow.

The combination of both consensus protocols helps to maintain a reliable communication and mitigate the zero-day attacks in IoT networks.

3.6. Context Management, Security Patching, and Reestablishment of Trust. The proposed approach relies on efficient decision-making over the context shared between the diagnosis systems. By analyzing the context, the proposed approach decides on rules for mitigating the threats of zero-day attacks. Further, it employs two protocols for alarming and data sharing between the correct devices across the network. However, with the proposed strategy, there are three major issues which should be handled for fault-free operations of the network in case a vulnerability is exploited. These include context management, security patching, and reestablishment of trust.

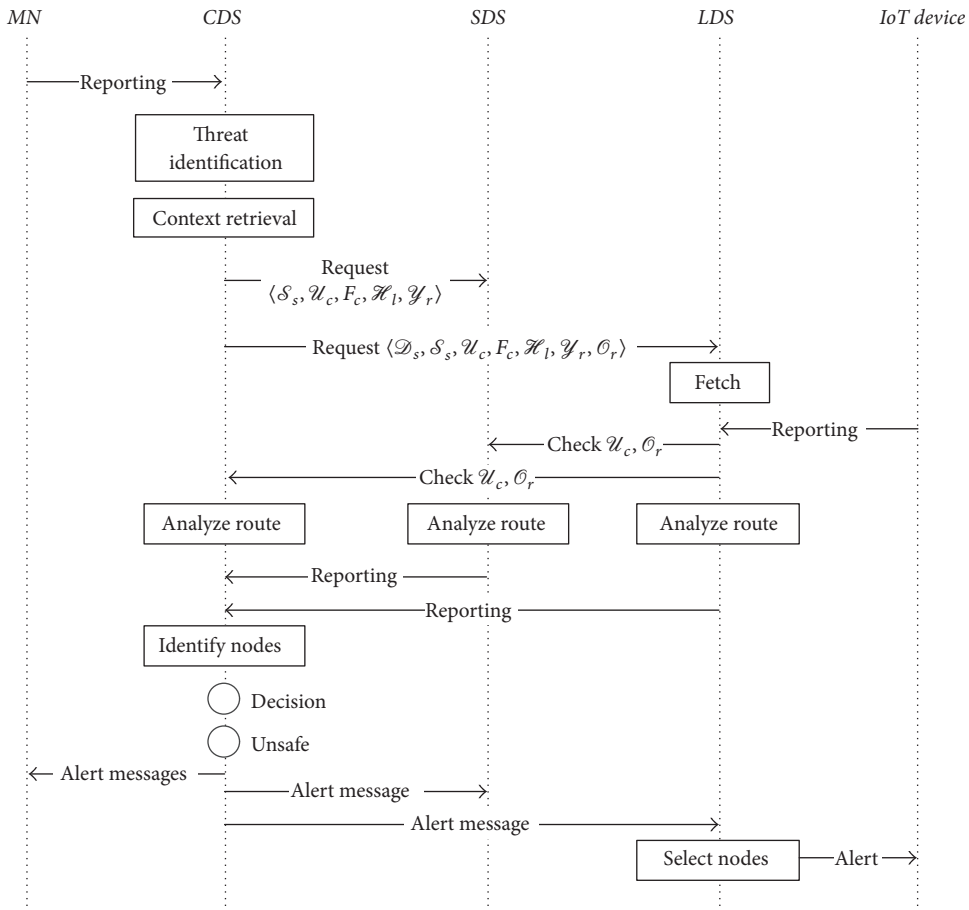


Figure 6: Consensus: alert protocol for zero-day attacks.

Context management is the responsibility of diagnosis systems and they use a secure channel for transmitting it across the network. Currently, the proposed approach is illustrated with a limited amount of context and decision solutions. This can be extended by an in-depth evaluation of IoT devices and generation of more contextual information which can help in identifying any device during regular operations. The context is updatable and can be modified with permissions of CDS as well as the trust establishing authorities. The context is stored as log files which are periodically updated and follows append mode which is dependent on a particular backup time for the replacement. SDS and LDS can also have a local storing point; however, this can overcome the issues of overheads involved in transmitting context across the network but raises issues related to local authentication as well as memory consumption. In the proposed approach, limited context is managed by the SDS and LDS and a majority of it is invoked directly from the device or CDS.

Once a vulnerability is exploited by a notorious group or a zero-day attack is launched, it is the responsibility of the proposed approach to counterfeit its effects and provide consensus rules for communication. After applicability of the proposed approach, it is the responsibility of the developers or network administrators to update the IoT firmware with new security patches using on-site or off-site mechanisms. On-site mechanisms are the traditional way of supporting an affected system, whereas, for off-site patching, the proposed approach can depend on the LDS or the SDS for reestablishing the trust and updating the firmware. However, the choice between the two is dependent on the fact whether a vulnerability is found or exploited. In the former case, off-site patching is successful, whereas, in the latter case, on-site patching is recommended. However, in any of the scenarios, the proposed approach can continue its mechanisms and support communication between the corrected devices without falling prey to the faulty nodes.

The final phase of mitigating a zero-day attack in IoT networks is the reestablishment of trust between the devices. Once security patching is done, the proposed approach operates as a counterpart and asks for context from the updated device. Since this is the first time after patching that the device is active, the default metrics are obtained from the security patches or the developers. By using default communications, the context is checked and a new pseudo ID is given to the recovered devices. Further, if the behavior of the device is found to be normal and aligned with the previous logs, the device is secured and its trust is reestablished. Since the proposed approach is iterative and continuously monitors the IoT devices as well as the connections between the MNs

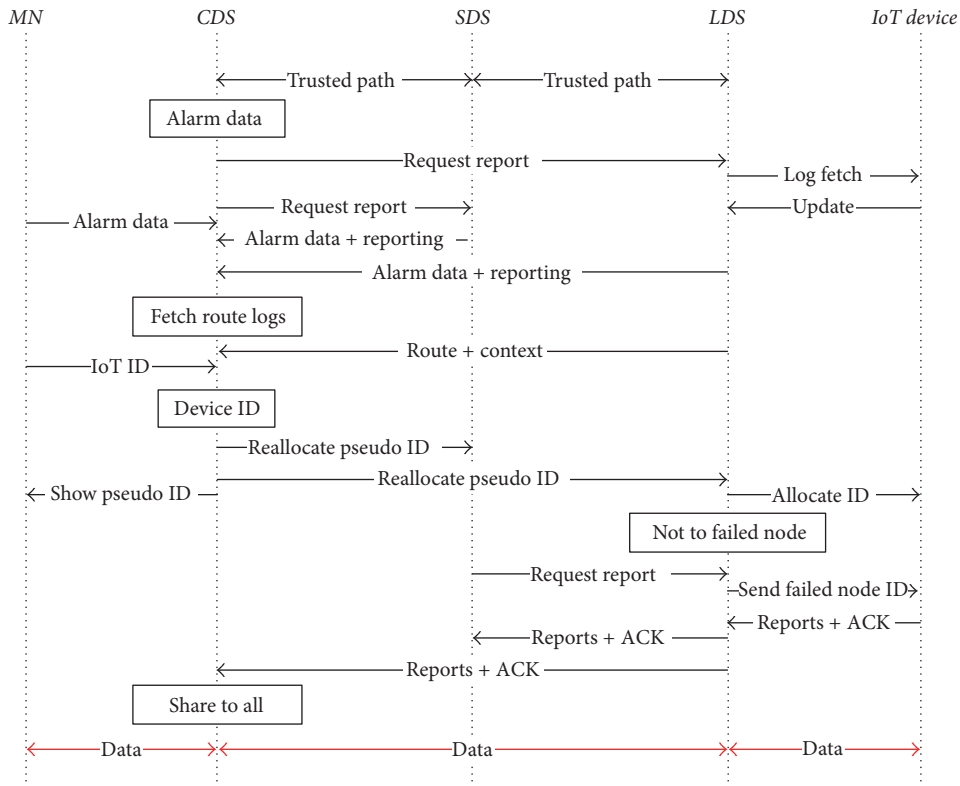


Figure 7: Consensus: critical data sharing protocol for zero-day attacks.

and the CNs, most possibilities of zero-day exploitations are counterfeited in a single attempt.

4. Performance Evaluations

The proposed approach is evaluated by using numerical simulations. The proposed approach is analyzed in two parts. The first part presents the network analysis by considering the zero-day attacks, and the second part presents the latency analysis for the two protocols proposed in this paper for mitigation of zero-day threats in IoT networks.

4.1. Network Analysis. These evaluations are performed in a numerically analyzed network by using *Matlab*™. A network with a single BS is considered which supports connectivity between MNs and CNs via APs. As described in the system model, the network is operated with three diagnosis systems with two protocols and an algorithm defined in the proposed approach. The network operates with 90% trustworthy components, the rest of which are under zero-day threat and may or may not lead to a full zero-day attack. This part of analysis presents the trends for reliability and consensus cost by varying the network size. Considering the parameter configurations given in Table 4, a randomized graph is formed between the network entities aiming at transmitting data between the MNs and the CNs.

First of all, the analysis is recorded for $\mathscr{R}$ against the variation in network size. The network size is the sum of all the nodes active in the network and possessing a connection with at least one of the MNs or the CNs. Figure 8 shows the reliability curve for the initial network and a network with and without infected nodes. The initial network curve serves as the baseline, and it can be analyzed that once the nodes with zero-day possibilities are removed from the communication by using the proposed approach, the value of $\mathscr{R}$ gradually increases and reaches a maximum where every possibility of zero-day threat is mitigated.

Table 4: Parameter configurations.

Parameter	Value	Description
$\lvert\mathscr{M}\rvert$	80	IoT devices
$\lvert\mathscr{H}\rvert$	2	HGWs
$\lvert\mathscr{B}\rvert$	1	BS
$\lvert\mathscr{N}\rvert$	10	MNs
$\lvert\mathscr{A}\rvert$	5	APs
$\lvert\mathscr{A}'\rvert$	2	Independent APs
$\mathscr{G} = (\mathscr{V}, \mathscr{E})$	100	Network graph
τ	90%	Trustworthy components
s	$\left((\tau)^4 \times \left(1 - \tau^{\tau-2}\right)\right) / (1 - \tau)$	Alternative routes [63]
α	0-1	Connectivity constant for $\mathscr{V}$
$\mathscr{T}$	5 ms	Halt time
$\mathscr{W}$	α	Link weight
η	2 per node	Instances per connection

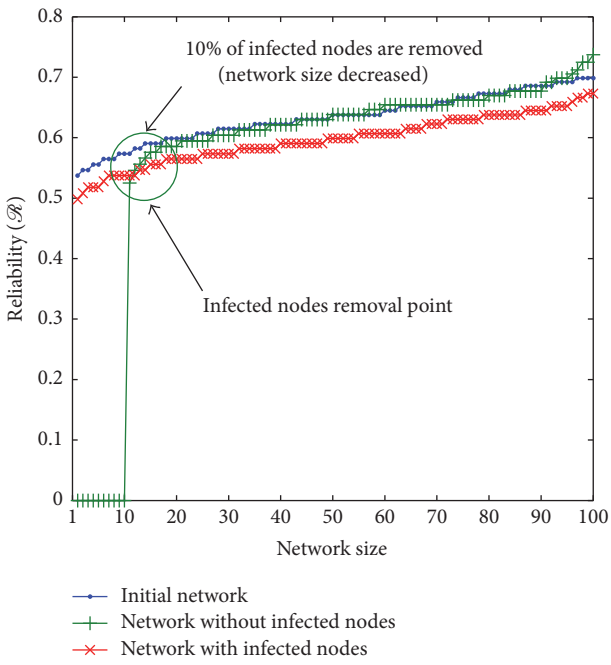


FIGURE 8: Reliability $\mathscr{R}$ versus network size.

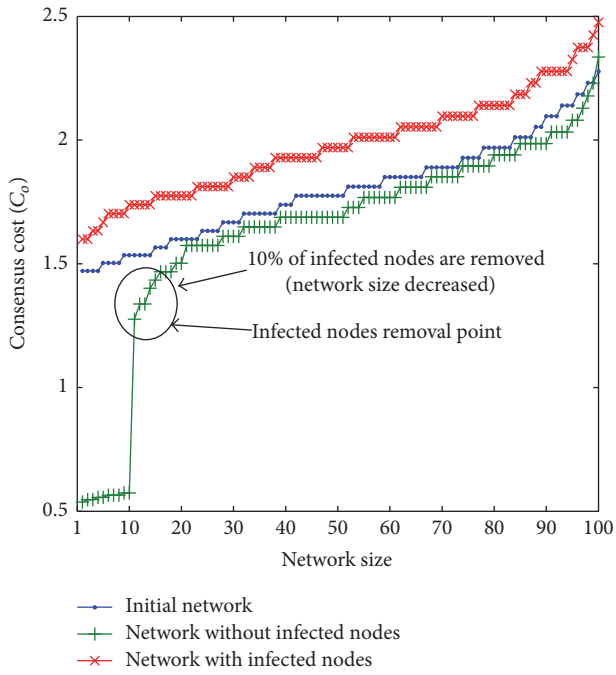


FIGURE 9: Consensus cost C_o versus network size.

The network can be operated with $\mathscr{R}_{\text{TH}}$ as discussed in the system model. Considering this figure, 0.5 can be treated as the threshold value for $\mathscr{R}_{\text{TH}}$. Below this value, a network cannot be considered reliable for communications and any value above it makes network safe despite the presence of infected nodes. It can be noticed that the network with infected nodes despite their identification has lesser reliability than the network with complete removal of such nodes. Thus, it is necessary to eliminate such nodes from the network as some traffic may still be passed to such nodes without expecting any forwarding mechanisms. This may cause excessive overheads and can further decrease the reliability of the network.

Contrary to $\mathscr{R}$, the consensus cost operates in a minimum way and the network with the removal of infected nodes has a better cost which is lower than the baseline as well as the network with infected nodes. Further, a network, which possesses infected nodes even after the identification, has higher values for C_o because infected nodes are also included while deriving actual cost for the network. These trends are presented in Figure 9. It is clear from the figure that, similar to $\mathscr{R}$ curves, it is important to operate a network with much knowledge allowing nontransmission of packets towards the infected nodes. Such behavior can be attained by the application of the proposed approach.

Figures 10 and 11 show values of $\mathscr{R}$ for baseline compared with attacker environment having 10% infected nodes and a network with the removal of the infected nodes, respectively. The red color in the former presents the lower values for reliability illustrating that the network which suffers from zero-day threats should be identified and moved towards their elimination as shown in the latter figure. Once the infected nodes are removed, as shown by a point of removal in Figure 11, the reliability of network can be managed and made comparable with the baseline.

Further, the network analysis is presented by statistical variations of the numerical results following $\mathscr{R}$ and C_o, as shown in Figures 12 and 13. Figure 12 shows the variation of network density versus $\mathscr{R}$. The curve fitting with a normal distribution of values shows the comparison for reliability values in baseline mode and in the network with and without infected nodes. Similar observations are seen for C_o in Figure 13. It can be seen that a network without infected nodes possesses lower values for consensus cost and higher values for reliability. However, reliability curves are dominated by the baseline, whereas the network with infected nodes dominates the consensus cost negatively. The behavior of these curves is aligned with (1)–(3).

The results presented in this section are able to justify that the proposed approach can maintain reliable communication despite the presence of infected nodes in the network. However, the success of the proposed approach depends on the number of alternative routes available for communication in a highly infected network. A network with extremely large effects of zero-day threats may possess a low value for $\mathscr{R}$, and such scenarios can lead to complete failure of the network. However, the alarming mechanisms of the proposed approach take care of such scenarios and prevent complete failure or shutdown of the network in extreme zero-day possibilities.

4.2. Latency Analysis. In the proposed approach, two different protocols are used for alert messaging as well as for data sharing during critical instances in the network. This section presents latency analysis for both the protocols and helps to understand the communication overheads caused due to

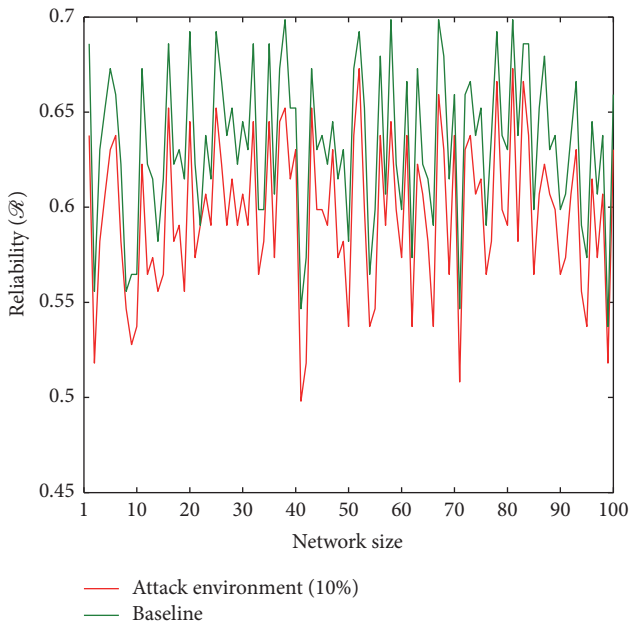


Figure 10: Reliability $\mathscr{R}$ attacker environment versus network size.

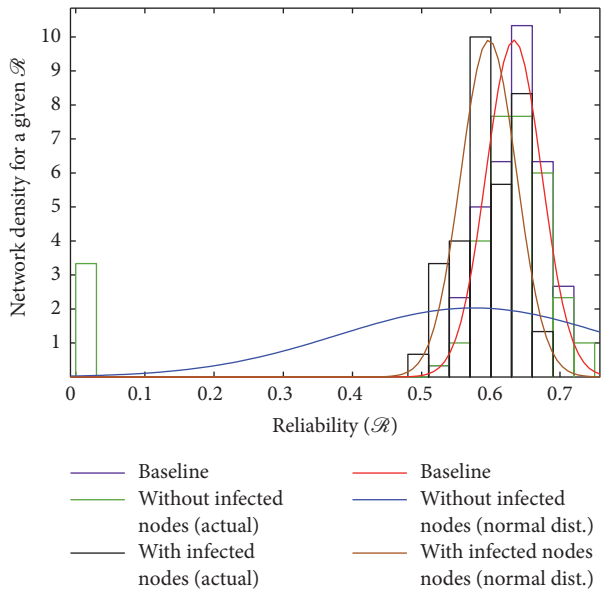


Figure 12: Statistical evaluation: network density for all values of $\mathscr{R}$ versus reliability $\mathscr{R}$.

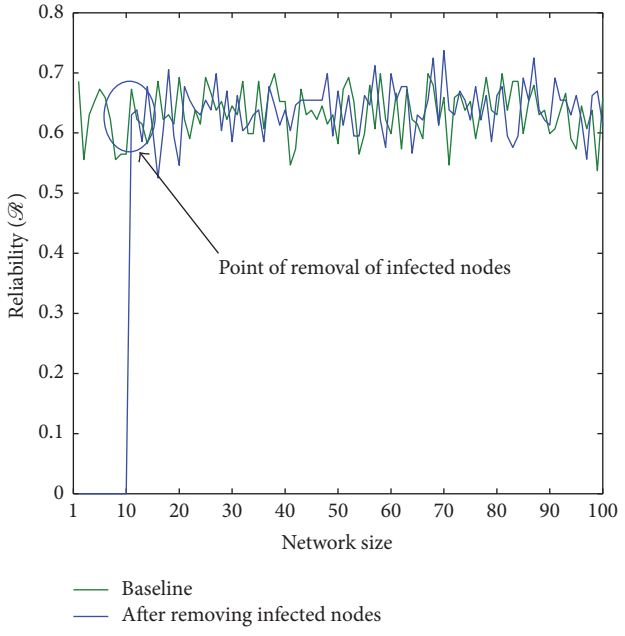


Figure 11: Reliability $\mathscr{R}$ without infected nodes versus network size.

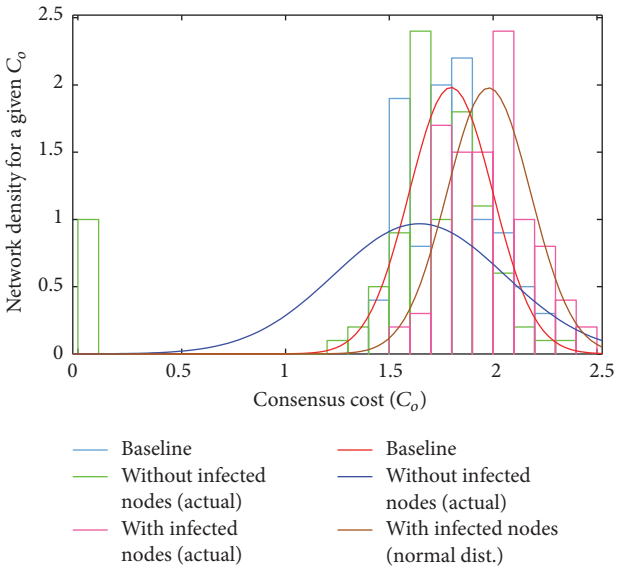


Figure 13: Statistical evaluation: network density for all values of C_o versus consensus cost C_o.

messaging between the network entities. Various metrics, their notations, and descriptions used in this section are given in Table 5. The results are observed by varying the network nodes in normal and infected scenarios. Similar to network analysis, the network size includes all the nodes involved in communication between the MNs and the CNs. The values for numerical analysis are used by observing the number of iterations required for generating a particular message. However, subjected to a particular environment, these values can be changed according to the configuration of the communication protocol used between the nodes.

The critical instances in both the protocols are marked by the identification of network nodes under a possible zero-day attack. For alert message protocol, the alert message generating latency $\mathscr{L}^A$ is calculated as

$$\mathscr{L}^A = 2\mathscr{T}_{\text{mn-cds}} + 3\mathscr{T}_{\text{cds-sds}} + 4\mathscr{T}_{cds\text{-lds}} + \mathscr{T}_{\text{sds-lds}} + 2\mathscr{T}_{\text{lds-iot}}, \quad (4)$$

and final alert message generating latency after decisions is calculated as

$$\mathscr{L}_f^A = \mathscr{L}^A + 4\mathscr{Q}_{\text{cds}} + \mathscr{Q}_{\text{sds}} + 3\mathscr{Q}_{\text{lds}}. \tag{5}$$

However, in networks with equal decision-making delay, (4) can be rewritten as

$$\mathscr{L}_f^A = \mathscr{L}^A + 8\mathscr{Q}_{\text{tds}}. \tag{6}$$

Now, for a wired link between the CDS, SDS, and LDS, one-way packet transport delay is calculated as [79, 80]

$$\mathscr{D}_e(\mathscr{P}, \mathscr{Z}) = \frac{\mathscr{P} \times \mathscr{Z}}{\beta} + \mathscr{D}_{\text{wired}}, \tag{7}$$

and, for wireless links, one-way packet transport delay is calculated as [79, 80]

$$\mathscr{D}_{\text{el}}(\mathscr{P}) = \mathscr{F}_s + (\mathscr{X} - 1)\, t. \tag{8}$$

Now, by using the above equations, the communication overheads for alert messaging can be calculated as

$$\begin{aligned}\mathscr{C}_{\text{overheads}} = {} & 2\mathscr{D}_{\text{el}}(\mathscr{P}_{\text{mn}}) + 2\mathscr{D}_{\text{el}}(\mathscr{P}_{\text{iot}}) \\ & + 3\mathscr{D}_e(\mathscr{P}, \mathscr{Z}_{\text{cds-sds}}) + \mathscr{D}_e(\mathscr{P}, \mathscr{Z}_{\text{sds-lds}}) \\ & + 4\mathscr{D}_e(\mathscr{P}, \mathscr{Z}_{\text{cds-lds}}) + \mathscr{L}_f^A.\end{aligned} \tag{9}$$

In the second part, the analysis is performed for critical data sharing protocol. For this, the message generation latency is calculated as

$$\begin{aligned}\mathscr{L}^C = {} & 3\mathscr{T}_{\text{mn-cds}} + 3\mathscr{T}_{\text{cds-sds}} + 2\mathscr{T}_{\text{sds-lds}} + 4\mathscr{T}_{\text{cds-lds}} \\ & + 5\mathscr{T}_{\text{lds-iot}},\end{aligned} \tag{10}$$

and final critical message generating latency after decisions is calculated as

$$\mathscr{L}_f^C = \mathscr{L}^C + 4\mathscr{Q}_{\text{cds}} + \mathscr{Q}_{\text{lds}}. \tag{11}$$

Now, similar to alert message protocol, with equal decision time, (11) can be rewritten as

$$\mathscr{L}_f^C = \mathscr{L}^C + 5\mathscr{Q}_{\text{tds}}. \tag{12}$$

Finally, the communication overheads for critical data sharing protocol can be calculated as

$$\begin{aligned}\mathscr{C}_{\text{overheads}}^{\mathscr{D}} = {} & 3\mathscr{D}_{\text{el}}(\mathscr{P}_{\text{mn}}) + 5\mathscr{D}_{\text{el}}(\mathscr{P}_{\text{iot}}) \\ & + 3\mathscr{D}_e(\mathscr{P}, \mathscr{Z}_{\text{cds-sds}}) \\ & + 2\mathscr{D}_e(\mathscr{P}, \mathscr{Z}_{\text{sds-lds}}) \\ & + 5\mathscr{D}_e(\mathscr{P}, \mathscr{Z}_{\text{cds-lds}}) + \mathscr{L}_f^C.\end{aligned} \tag{13}$$

Now, by using the above-defined analysis model and the values from Table 5, results are observed for message latency as well as communication overheads. The numerical and timing analyses suggest that the proposed approach is capable of handling zero-day attack scenarios efficiently as a minimal difference is observed for alert message latency, as shown in Figure 14. It is evident from this graph that the proposed approach can handle network under zero-day attack with sufficient efficiency as the variation between the baseline and the actual observed values during attacks for alert message protocol is negligible. This graph shows results for $\mathscr{L}^A$ and $\mathscr{L}_f^A$ in the baseline as well as infected mode.

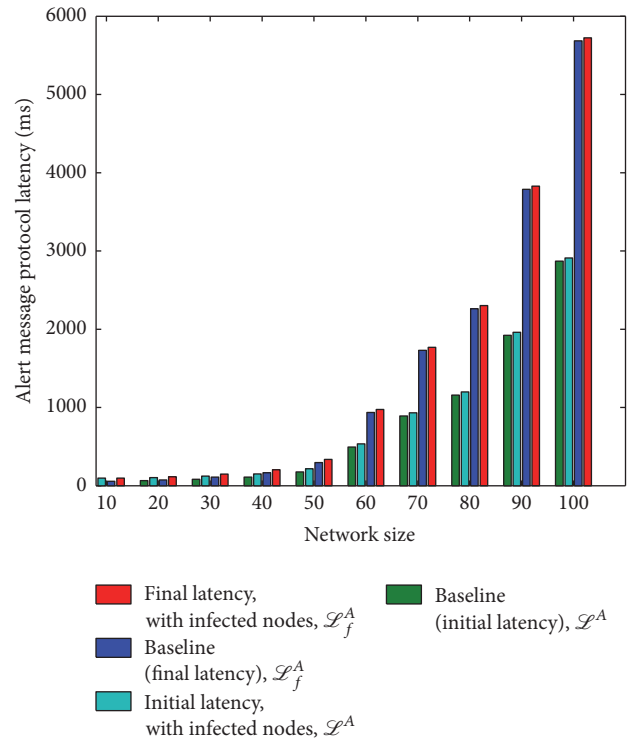


Figure 14: Alert message protocol latency versus network size.

Similarly, these results are recorded for analyzing the message latency for critical data sharing protocols by following the formulations in (10) and (12). The less difference between the message latency in the baseline and that in the infected scenario suggests the efficient operations of the proposed approach. These trends can be visualized in Figure 15. Despite the fact that message latency increases with an increase in the network size, the overall difference between the actual scenario and the observed scenario is less; thus, the proposed approach can be used for performance-based reliable communication during mitigation of zero-day attacks in IoT networks.

Finally, the overall performance of the proposed approach in different modes with details on communication overheads can be observed in Figure 16. This graph presents results for $\mathscr{C}_{\text{overheads}}$ and $\mathscr{C}_{\text{overheads}}^{\mathscr{D}}$. The graph considers variation in one-way transport delay which is affected by the frame error rate. The values for F_s are observed by varying frame error rate between 0 and 0.9 on a stepping scale of 0.1.

The results shown in this figure suggest that the overheads increase with an increase in one-way transport delay, and

Table 5: Latency analysis (notations).

Notations	Values	Description
$\mathscr{L}^A$	TBC	Alert message latency
$\mathscr{L}_f^A$	TBC	Final/total alert message latency
$\mathscr{T}_{\text{mn-cds}}$	$\delta \times \lvert\mathscr{N}\rvert \times \lvert\mathscr{M}\rvert \times \lvert\mathscr{B}\rvert \times \lvert\mathscr{A}\rvert \times \lvert\mathscr{K}\rvert$	Evaluation time between MN and CDS
δ	Exponential: 1–30 packets	Traffic arrival time
$\mathscr{T}_{\text{cds-sds}}$	5 ms	Evaluation time between CDS and SDS
$\mathscr{T}_{\text{cds-lds}}$	5 ms	Evaluation time between CDS and LDS
$\mathscr{T}_{\text{sds-lds}}$	10 ms	Evaluation time between SDS and LDS
$\mathscr{T}_{\text{lds-iot}}$	10 ms	Evaluation time between LDS and IoT
$\mathscr{Q}_{\text{tds}}$	5 ms	Average decision time per component
$\mathscr{Q}_{\text{cds}}$	5 ms	Decision time for CDS
$\mathscr{Q}_{\text{sds}}$	5 ms	Decision time for SDS
$\mathscr{Q}_{\text{lds}}$	5 ms	Decision time for LDS
$\mathscr{P}$	256 bytes	Message size
$\mathscr{F}_s$	$\mathscr{D}_o * (1 - \omega)$	One-way frame transport delay
ω	0–0.9, step 0.1	Frame error
$\mathscr{Z}$	10–100	Number of hops
β	10 MHz	Bandwidth
$\mathscr{D}_{\text{wired}}$,$\mathscr{D}_o$	20 ms	Delay
$\mathscr{X}$	0.5	Ratio packet size to frame size
t	20 ms	Interframe time
$\mathscr{Z}_{\text{cds-sds}}$	10	Hop distance between CDS and SDS
$\mathscr{Z}_{\text{cds-lds}}$	10	Hop distance between CDS and LDS
$\mathscr{Z}_{\text{sds-lds}}$	10	Hop distance between SDS and LDS
$\mathscr{L}^C$	TBC	Critical message latency
$\mathscr{L}_f^C$	TBC	Final/total critical message latency

*TBC: to be calculated with formula in Section 4.2.

this increase can also be observed for the baseline scenarios. More overheads are observed in the proposed approach when a network is under extreme zero-day attack. However, considering the level of reliability provided by the proposed approach during the presence of attackers, these results can be considered efficient for IoT networks. The excessive overheads are caused by security patching procedures, regeneration of alert messages, and reestablishment of trust. Out of these, regeneration of alert messages can be ignored, which will definitely affect the performance of the proposed approach on the better side. However, a network with a lower rate than a network with complete failure is much desirable in highly critical scenarios. Thus, considering the results and performance evaluations, it can be concluded that the proposed approach can mitigate the zero-day attacks without much effect on the performance of uninfected nodes.

To the best of our knowledge, there are no competitive approaches for IoT networks which can handle zero-day attacks simultaneously with reliable communication. However, on the basis of relativity, some of the recently proposed state-of-the-art solutions are used for comparison with the proposed solution as shown in Table 6. The existing solutions are efficient in detecting vulnerabilities and possibilities leading to zero-day attacks in IoT networks. However, these solutions do not emphasize much on alerting the connected nodes for the identified attack in the network. Further, these solutions do not show any sufficient mechanism for handling communication once an attack is identified. Thus, with these analyses, it is evident that the proposed approach can serve as a benchmark solution for real-time mitigation of zero-day attacks along with reliable communication in IoT networks.

5. Conclusions

"Zero-day" vulnerabilities and attacks are highly critical for IoT networks. These attacks can affect IoT devices by exploiting the defense perimeter of their network. In this paper, a detailed study was presented on zero-day attacks in IoT networks. Next, a consensus framework was proposed for mitigation of zero-day attacks in IoT networks. The proposed approach used context-behavior of IoT devices as a detection mechanism followed by alert message protocol and critical data sharing protocol for reliable communication during attack mitigation. The proposed protocol was evaluated numerically and the results suggested that the

Table 6: Comparative analysis with state-of-the-art solutions.

Approach	Authors	Network type	Zero-day detection	Scalability	Overheads	Latency	Response time	Reliable communication	Alert messages	Usability
DFA-AD	Sharma et al. [64]	IoT	Yes	—	—	Medium	—	No	No	Distributed
Self-protecting computing systems	Chen et al. [65]	IoT	Yes	—	Low	Low	—	No	No	Centralized
Hybrid layered architecture	Singh et al. [66]	General	Yes	—	—	Medium	—	No	No	Centralized
Proposed	Sharma et al.	IoT	Yes	High	Low	Low	Low	Yes/high	Yes	Distributed

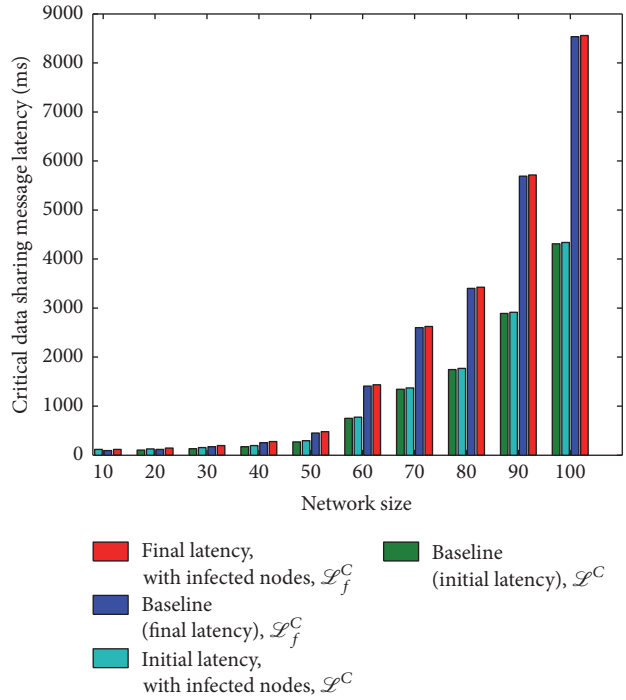


Figure 15: Critical data sharing protocol message latency versus network size.

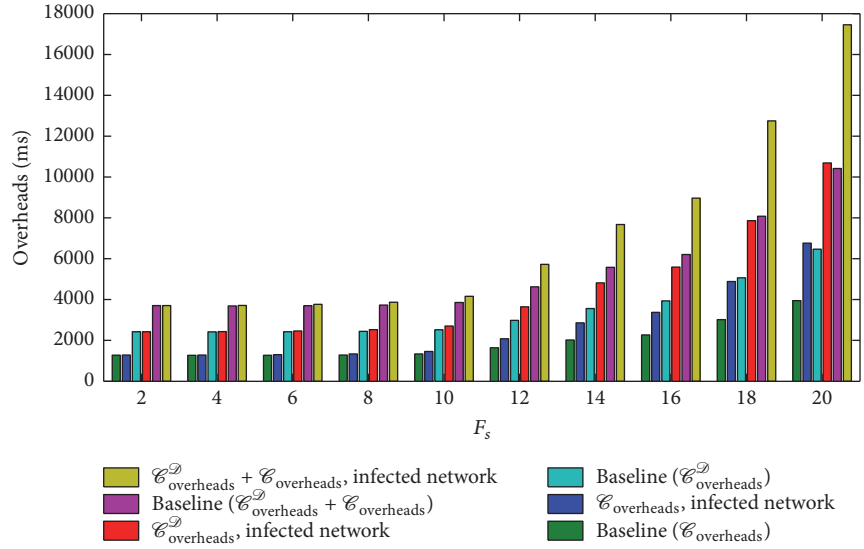


Figure 16: Communication overheads for alert message protocol and critical data sharing protocol versus one-way frame transport delay F_s.

proposed approach can serve the purpose of detection and elimination of zero-day attacks in IoT network without compromising its performance. A state-of-the-art comparison was also presented that justified the performance as well as the benchmark standard fixed by the proposed approach. The results for latency and overheads suggest high-performance network formation even in the presence of zero-day attacks.

In the future, the focus will be given on securing the passes of alert message protocol and critical data sharing protocol by investigating them with different security mechanisms. Further, the proposed approach will be analyzed by considering the actual behavior and message size for different IoT devices.

Disclosure

A part of this paper was presented at a Conference on Information Security and Cryptography (CISC-S 17), June 22-23, 2017, Asan, South Korea.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) that is funded by the Ministry of Education (NRF-2015R1D1A1A01057300) and in part by the Soonchunhyang University Research Fund.

References

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

[2] V. Sharma, J. D. Lim, J. N. Kim, and I. You, "SACA: Self-Aware Communication Architecture for IoT Using Mobile Fog Servers," *Mobile Information Systems*, vol. 2017, pp. 1–17, 2017.

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

[4] V. Desnitsky, D. Levshun, A. Chechulin, and I. Kotenko, "Design technique for secure embedded devices: Application for creation of integrated cyber-physical security system," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 7, no. 2, pp. 60–80, 2016.

[5] I. You and K. Yim, "Malware obfuscation techniques: a brief survey," in *Proceedings of the 5th International Conference on Broadband Wireless Computing, Communication and Applications (BWCCA '10)*, pp. 297–300, November 2010.

[6] I. Kotenko and A. Chechulin, "Attack modeling and security evaluation in siem systems," *International Transactions on Systems Science and Applications*, vol. 8, pp. 129–147, 2012.

[7] M. J. Covington and R. Carskadden, "Threat implications of the Internet of Things," in *Proceedings of the Cyber Conflict (CyCon), 2013 5th International Conference*, pp. 1–12, 2013.

[8] I. Agrafiotis, A. Erola, M. Goldsmith, and S. Creese, "Formalising policies for insider-threat detection: A tripwire grammar," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 8, no. 1, pp. 26–43, 2017.

[9] F. Kammüller, M. Kerber, and C. W. Probst, "Insider threats and auctions: Formalization, mechanized proof, and code generation," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 8, no. 1, pp. 44–78, 2017.

[10] P. Firstbrook and N. MacDonald, *A buyers guide to endpoint protection platforms*, https://www.gartner.com/doc/2973617/buyers-guide-endpoint-protection-platforms.

[11] R. Tian, L. Batten, R. Islam, and S. Versteeg, "An automated classification system based on the strings of trojan and virus families," in *Proceedings of the 2009 4th International Conference on Malicious and Unwanted Software, MALWARE 2009*, pp. 23–30, can, October 2009.

[12] D. Bilar, "Opcodes as predictor for malware," *International Journal of Electronic Security and Digital Forensics*, vol. 1, pp. 156–168, 2007.

[13] G. Bonfante, M. Kaczmarek, and J.-Y. Marion, "Morphological detection of malware," in *Proceedings of the 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008*, pp. 1–8, usa, October 2008.

[14] A. Pektas, M. Eris, and T. Acarman, "Proposal of n-gram based algorithm for malware classification," in *in Proceedings of the Fifth International Conference on Emerging Security Information, Systems and Technologies*, pp. 21–27, French Riviera, France, 2011.

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

[16] M. Egele, E. Kirda, and C. Kruegel, "Mitigating drive-by download attacks: Challenges and open problems," *IFIP Advances in Information and Communication Technology*, vol. 309, pp. 52–62, 2009.

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

[18] Z. Ruili, P. Jianfeng, T. Xiaobin, and X. Hongsheng, "Application of CLIPS expert system to malware detection system," in *Proceedings of the 2008 International Conference on Computational Intelligence and Security, CIS 2008*, pp. 309–314, chn, December 2008.

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

[20] T. Dube, R. Raines, G. Peterson, K. Bauer, M. Grimaila, and S. Rogers, "Malware target recognition via static heuristics," *Computers & Security*, vol. 31, no. 1, pp. 137–147, 2012.

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

[22] H. Lu, X. Wang, B. Zhao, F. Wang, and J. Su, "ENDMal: An anti-obfuscation and collaborative malware detection system using syscall sequences," *Mathematical and Computer Modelling*, vol. 58, no. 5-6, pp. 1140–1154, 2013.

[23] C. Williams, "Applications of genetic algorithms to malware detection and creation," 2009.

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

[25] F. Bao, I. Chen, and J. Guo, "Scalable, adaptive and survivable trust management for community of interest based Internet of Things systems," in *Proceedings of the 2013 IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, pp. 1–7, Mexico City, Mexico, March 2013.

[26] D. Chen, G. Chang, and D. Sun, "TRM-IoT: a trust management model based on fuzzy reputation for internet of things," *Computer Science and Information Systems*, vol. 8, no. 4, pp. 1207–1228, 2011.

[27] R. Neisse, G. Steri, I. N. Fovino, and G. Baldini, "SecKit: a model-based security toolkit for the internet of things," *Computers & Security*, vol. 54, pp. 60–76, 2015.

[28] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi, "The cluster between internet of things and social networks: Review

and research challenges," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 206–215, 2014.

[29] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, 2013.

[30] O. L. Barakat, A. R. Ramli, F. Hashim, K. Samsudin, I. A. Albaltah, and M. M. Al-Habshi, "Scarecrow: Scalable malware reporting, detection and analysis," *Journal of Convergence Information Technology*, vol. 8, no. 14, p. 1, 2013.

[31] Y. Qiao, Y. Yang, L. Ji, and J. He, "Analyzing malware by abstracting the frequent itemsets in API call sequences," in *Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '13)*, pp. 265–270, July 2013.

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

[33] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 646–656, 2013.

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

[35] L. Feng, X. Liao, Q. Han, and H. Li, "Dynamical analysis and control strategies on malware propagation model," *Applied Mathematical Modelling*, vol. 37, no. 16-17, pp. 8225–8236, 2013.

[36] D. Debarr, V. Ramanathan, and H. Wechsler, "Phishing detection using traffic behavior, spectral clustering, and random forests," in *Proceedings of the 11th IEEE International Conference on Intelligence and Security Informatics, IEEE ISI 2013*, pp. 67–72, usa, June 2013.

[37] M. Scheutz and V. Andronache, "Architectural mechanisms for dynamic changes of behavior selection strategies in behavior-based systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 6, pp. 2377–2395, 2004.

[38] A. P. Lauf, R. A. Peters, and W. H. Robinson, "Embedded intelligent intrusion detection: A behavior-based approach," in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops/Symposia, AINAW'07*, pp. 816–821, can, May 2007.

[39] J. Hu, "Host-based anomaly intrusion detection," *Handbook of Information and Communication Security*, pp. 235–255, 2010.

[40] A. S. Ashoor and S. Gore, "Intrusion detection system: case study," in *in Proceedings of the International Conference on Advanced Materials Engineering*, vol. 15, pp. 6–9, 2011.

[41] S. S. Murtaza, W. Khreich, A. Hamou-Lhadj, and M. Couture, "A host-based anomaly detection approach by representing system calls as states of kernel modules," in *Proceedings of the 2013 IEEE 24th International Symposium on Software Reliability Engineering, ISSRE 2013*, pp. 431–440, usa, November 2013.

[42] H. Kaur and N. Gill, "Host based Anomaly Detection using Fuzzy Genetic Approach (FGA)," *International Journal of Computer Applications*, vol. 74, no. 20, pp. 5–9, 2013.

[43] S. Cesare and Y. Xiang, "A fast flowgraph based classification system for packed and polymorphic malware on the endhost," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA '10)*, pp. 721–728, Perth, Australia, April 2010.

[44] D. Song, D. Brumley, and H. Yin, "BitBlaze: a new approach to computer security via binary analysis," in *The Journal of Information System Security*, vol. 5352 of *Lecture Notes in Computer Science*, pp. 1–25, Springer, Berlin, Germany, 2008.

[45] K. Yoshioka, Y. Hosobuchi, T. Orii, and T. Matsumoto, "Vulnerability in public malware sandbox analysis systems," in *Proceedings of the 2010 10th Annual International Symposium on Applications and the Internet, SAINT 2010*, pp. 265–268, kor, July 2010.

[46] G. Willems, T. Holz, and F. Freiling, "Toward automated dynamic malware analysis using CWSandbox," *IEEE Security & Privacy*, vol. 5, no. 2, pp. 32–39, 2007.

[47] D. Inoue, K. Yoshioka, M. Eto, Y. Hoshizawa, and K. Nakao, "Automated malware analysis system and its sandbox for revealing malware's internal and external activities," *IEICE Transaction on Information and Systems*, vol. E92-D, no. 5, pp. 945–954, 2009.

[48] S. Miwa, T. Miyachi, M. Eto, M. Yoshizumi, and Y. Shinoda, "Design and implementation of an isolated sandbox with mimetic internet used to analyze malwares," in *DETER*, 2007.

[49] W. Jiang, R. Wang, Z. Xu, Y. Huang, S. Chang, and Z. Qin, "PRUB: A Privacy Protection Friend Recommendation System Based on User Behavior," *Mathematical Problems in Engineering*, vol. 2016, Article ID 8575187, 2016.

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

[51] V. Sharma, I. You, and R. Kumar, "ISMA: Intelligent Sensing Model for Anomalies Detection in Cross Platform OSNs With a Case Study on IoT," *IEEE Access*, vol. 5, pp. 3284–3301, 2017.

[52] R. Kaur and M. Singh, "A survey on zero-day polymorphic worm detection techniques," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1520–1549, 2014.

[53] R. Kaur and M. G. Singh, *Efficient Zero-day Attacks Detection Techniques [Ph.D. thesis]*, 2016.

[54] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network Diversity: A Security Metric for Evaluating the Resilience of Networks Against Zero-Day Attacks," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 5, pp. 1071–1086, 2016.

[55] W. House, "Trustworthy cyberspace: Strategic plan for the federal cybersecurity research and development program," *Report of the National Science and Technology Council, Executive Office of the President*, 2011.

[56] S. Viswanathan, R. Tan, and D. K. Y. Yau, "Exploiting Power Grid for Accurate and Secure Clock Synchronization in Industrial IoT," in *Proceedings of the 2016 IEEE Real-Time Systems Symposium, RTSS 2016*, pp. 146–156, prt, December 2016.

[57] K. Sonar and H. Upadhyay, "A survey: Ddos attack on internet of things," *International Journal of Engineering Research and Development*, vol. 10, no. 11, pp. 58–63, 2014.

[58] A. Sajid, H. Abbas, and K. Saleem, "Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges," *IEEE Access*, vol. 4, pp. 1375–1384, 2016.

[59] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[60] T.-H. Cho and G.-M. Jeon, "A method for detecting man-in-the-middle attacks using time synchronization one time password in interlock protocol based internet of things," *Journal of Applied and Physical Sciences*, vol. 2, no. 2, pp. 37–41, 2016.

[61] J. Liu, Y. Xiao, and C. P. Chen, "Authentication and access control in the Internet of things," in *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW '12)*, pp. 588–592, IEEE, Macau, China, June 2012.

[62] S. Na, D. Hwang, W. Shin, and K.-H. Kim, "Scenario and countermeasure for replay attack using join request messages in lorawan," in *Proceedings of the Information Networking (ICOIN, 2017 International Conference*, pp. 718–720, 2017.

[63] J. L. Martin, "Complete graphs," https://www.math.ku.edu/jmartin/courses/math105-F11/Lectures/chapter6-part2.pdf.

[64] P. K. Sharma, S. Y. Moon, D. Moon, and J. H. Park, "DFA-AD: a distributed framework architecture for the detection of advanced persistent threats," *Cluster Computing*, vol. 20, no. 1, pp. 597–609, 2017.

[65] Q. Chen, S. Abdelwahed, and A. Erradi, "A model-based validated autonomic approach to self-protect computing systems," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 446–460, 2014.

[66] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "WITHDRAWN: A hybrid layered architecture for detection and analysis of network based Zero-day attack," *Computer Communications*, vol. 106, pp. 100–106, 2017.

[67] D. Kim, Y. Kim, J. Moon, S.-J. Cho, J. Woo, and I. You, "Identifying windows installer package files for detection of pirated software," in *Proceedings of the 8th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2014*, pp. 287–290, gbr, July 2014.

[68] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[69] G. Portokalidis, A. Slowinska, and H. Bos, "Argos: An emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation," in *Proceedings of the 2006 EuroSys Conference*, pp. 15–27, bel, April 2006.

[70] K. Palani, E. Holt, and S. Smith, "Invisible and forgotten: Zero-day blooms in the IoT," in *Proceedings of the 13th IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2016*, aus, March 2016.

[71] B. Wanswett and H. K. Kalita, "The Threat of Obfuscated Zero Day Polymorphic Malwares: An Analysis," in *Proceedings of the 7th International Conference on Computational Intelligence and Communication Networks, CICN 2015*, pp. 1188–1193, ind, December 2015.

[72] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012*, pp. 833–844, usa, October 2012.

[73] J. B. Fraley and J. Cannady, "The promise of machine learning in cybersecurity," in *Proceedings of the SoutheastCon 2017*, pp. 1–6, Concord, NC, USA, March 2017.

[74] P. Duessel, C. Gehl, U. Flegel, S. Dietrich, and M. Meier, "Detecting zero-day attacks using context-aware anomaly detection at the application-layer," *International Journal of Information Security*, pp. 1–16, 2016.

[75] S. Chamotra, R. K. Sehgal, and R. S. Misra, "Honeypot Baselining for Zero Day Attack Detection," *International Journal of Information Security and Privacy*, vol. 11, no. 3, pp. 63–74, 2017.

[76] X. Sun, J. Dai, P. Liu, A. Singhal, and J. Yen, "Towards probabilistic identification of zero-day attack paths," in *Proceedings of the 2016 IEEE Conference on Communications and Network Security, CNS 2016*, pp. 64–72, usa, October 2016.

[77] W. Haider, G. Creech, Y. Xie, and J. Hu, "Windows based data sets for evaluation of robustness of Host based Intrusion Detection Systems (IDS) to zero-day and stealth attacks," *Future Internet*, vol. 8, no. 3, article no. 29, 2016.

[78] D. W. Coit and A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Transactions on Reliability*, vol. 45, no. 2, pp. 254–263, 1996.

[79] I. You and J. Lee, "SPFP: Ticket-based secure handover for fast proxy mobile IPv6 in 5G networks," *Computer Networks*, 2017.

[80] J.-H. Lee, J.-M. Bonnin, I. You, and T.-M. Chung, "Comparative handover performance analysis of IPv6 mobility management protocols," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 3, pp. 1077–1088, 2013.