IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON INTERNET-OF-THINGS ATTACKS AND DEFENSES: RECENT ADVANCES AND CHALLENGES

Received July 18, 2020, accepted August 5, 2020, date of publication August 17, 2020, date of current version October 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3017009

Prioritization and Alert Fusion in Distributed IoT Sensors Using Kademlia Based Distributed Hash Tables

MANSOOR NASIR[1], KHAN MUHAMMAD[2,4], (Member, IEEE), PAOLO BELLAVISTA[3], (Senior Member, IEEE), MI YOUNG LEE[2], AND MUHAMMAD SAJJAD[1]

[1]Department of Computer Science, Islamia College University Peshawar, Peshawar 25000, Pakistan
[2]Intelligent Media Laboratory, Digital Contents Research Institute, Sejong University, Seoul 143-747, South Korea
[3]Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy
[4]Department of Software, Sejong University, Seoul 143-747, South Korea

Corresponding authors: Khan Muhammad (khan.muhammad@ieee.org) and Muhammad Sajjad (muhammad.sajjad@icp.edu.pk)

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1B07043302).

ABSTRACT Distributed intrusion detection systems (IDS) are primarily deployed across the network to monitor, detect, and report anomalies, as well as to respond in real-time. Predominantly, an IDS is equipped with a set of rules that it needs to infer to be able to perform efficient detection. However, reducing the generation of false alarms is a major challenge in any IDS implementation. Additionally, the sheer number of IoT devices that generate alarms in a moderately large sensor network may be overwhelming. In order to reduce alarms, this paper contributes to the field by proposing an original framework that limits the number of generated messages without compromising detection accuracy. The primary idea is to exploit mid-level nodes called collectors where similar alerts are collected and analyzed independently. Priority is assigned to each alert and similar alerts are fused to respective collectors for more informed decision making. In addition, Kademlia based Distributed Hash Table (DHT) is used for efficient alert transportation and distributed fusion of similar alerts. In order to minimize false alarm rate, event correlation is used to find similarity between events fused by different detection sensors. The framework is implemented in a fog-based environment to assess and evaluate the efficiency of the proposed system in edge network. The architecture is evaluated with the recognized DARPA 1999 dataset; the reported results show that the proposed technique reduces message generation by 62% while achieving false positive accuracy over 80%.

INDEX TERMS Alert correlation, alert fusion, distributed hash tables, intrusion detection.

I. INTRODUCTION

Network traffic anomalies are very common these days and identifying them not only quickly but more efficiently is essential especially for large networks and service providers. Related annual losses and online breaches are getting higher every year. It is essential for large companies to stay available at all time without any disruption to the clients. Coordinated attacks on large scale networks pose a major threat to network infrastructure around the world. Attacks like Port Scan and Distributed Denial of Service (DDoS) can affect multiple networks at the same time. In July 2018, a malware named ''Emotet'' has been exposed and has shown to be very destructive and costly. As of January 2019 it is one of the most dangerous malware on the Internet, representing 19% of the overall malware discovered [1]. According to [2] in the first three quarters of 2019 the number of malware attacks launched was 7.2 billion, among which 151.9 million was ransomware attacks. The number of these attacks is decreasing; however, the attacks are getting more nefarious and are evading commonly used sandbox technologies. It is important to mention here that the number of attacks in the IoT domain is increased to 25 million, i.e., around 33% more than in 2018. This highlights even more the fact that securing IoT applications is paramount and absolutely necessary.

Large scale attacks are generally hard to detect, because the evidence of these attacks is spread across multiple hosts. In order to detect such attacks, we first have to collect

The associate editor coordinating the review of this manuscript and approving it for publication was Weizhi Meng.

pieces of information from multiple hosts and then aggregate and correlate them to reach an accurate conclusion. In Collaborative Intrusion Detection Systems (CIDS) evidences from multiple detection sensors are collected and analyzed at some higher-level nodes [3]. However, combining information from heterogeneous sensors pose several problems such as: 1) Huge amount of messages generated by detection sensors; 2) Pre-processing, filtering [4], and unifying alert information; 3) Sharing alert information by using a common representation; 4) Aggregating and correlating pieces of evidences; and 5) Evolvability and changes in attack mechanism.

Smart cities are not a new concept anymore, future cities will be equipped with state-of-the-art cyber security system with automated decision-making capabilities in different areas i.e. healthcare [5]. IoT applications are being proposed by many researchers in the recent past. The development in the field of IoT is staggering and the adaptation of IoT in different areas i.e., home automation and intelligent transportation system pave way for different related areas. The rise in IoT applications introduces billions of connected IoT devices and securing each one of them is a technical challenge. Detecting network traffic anomalies is challenging as there are many variables to take into account in very large-scale deployment environments. No matter how small the anomaly may be, collecting it, is very important, as it can help correlating and aggregating events more accurately. A similar work has been proposed recently, where Li *et al.* [6], proposed a new architecture to enhance the sensitivity of detectors. The approach makes use of collective feedback and alarm aggregation along with trust management between nodes.

Reviewing the recent advancements in the field paves the way for us to recognize that the sheer amount of related events produced by sensors requires the following features for an innovative IDS infrastructure: 1) Efficient routing of similar alarms to respective intermediate collectors without the consultation of some centralized directory or without flooding the entire network; 2) Support to some kind of querying language that can collect and aggregate information from distributed nodes and multiple levels of collectors; 3) Support to effective load balancing; 4) Fault tolerance, e.g., efficient management of node join/leave in a distributed environment; and 5) Generation and consideration of only enough relevant information, so that collectors are not overwhelmed or slowed down by data with scarce meaning or impact.

An effective IDS faces many challenges including the management of large volumes of alert data generated by sensor nodes. Moreover, the underlying understanding of alerts is essential as singular instance of the alert may seem harmless but may be part of a large-scale attack. One more aspect that is usually neglected is the response to the detected threats: the IDS deployments need to carefully identify alert severity and to respond accordingly without compromising the user experience. By far one of the most difficult challenges that any IDS need to address is its effective and efficient implementation in a real-world environment with limited resources for analysis and visualization.

Keeping all the above challenges in view, we propose an original IDS framework based on Distributed Hash Table (DHT) that effectively provides decentralized routing. We also present a novel prioritization scheme to reduce alerts generated by sensors in a collaborative distributed IDS (DIDS). DHT is often used for Peer-to-Peer file sharing [7], content distribution over wide-scale networks [8], and even for streaming contents [9].

To the best of our knowledge, the solution in this paper is the first proposal that synergically exploits, together, prioritization schemes and distributed alert fusion and aggregation by using DHT. In particular, we believe that this paper primarily contributes to the literature in the field along the following guidelines:

- We designed a framework for alert reduction by introducing priority to each alert. Priority is assigned by dynamically evaluating alert and using several evaluation metrics including severity, confidence, correlation, and service history.
- In order to route alert information, the Kademlia topology of DHT is used as we prove that it is more efficient than previously used Chord and is also DDoS-resilient. This also favors load balancing among participating peers and has demonstrated to be less costly on each node join/leave.
- Our proposal is thoroughly evaluated against the recognized DARPA 1999 dataset and the results show that the proposed system is effective in reducing number of messages while maintaining an appropriate false-positive rate.

In Section II and III, we provide the readers with the necessary background and discuss the currently employed techniques for DIDS. In Section IV, we present the architecture of our IDS framework based on the Kademlia-assisted DHT overlay network. Results analysis and discussions are highlighted in Section V, and finally the paper is concluded in Section VI.

II. BACKGROUND

Adversaries are using different techniques to penetrate the network by exploiting vulnerabilities and loop holes. The nature of attack usually depends on the motives behind the attack but, no matter how complex the attack is, there are always some evidence left behind, that can be back tracked to the origin of the attack. Details of primary terms used in this paper are given in Table 1.

In a distributed environment different detection sensor are deployed to collect alert information for the purpose of correlation and scenario reconstruction [10]. Centralized approach of analysis gives more control over the participating nodes but due to single point of failure, performance and scalability issues, it is rarely used in distributed environment. The main challenge that any collaborative IDS faces is the maintenance of trust between different nodes. Alexopoulos *et al.* [11],

TABLE 1. The used terminology.

Term	Description
Intrusion	An attempt to break into some unauthorized system for the purpose of misuse
Intruder	The person or bot attempting the intrusion
Alert	A single detection of an intrusion
Event	Subsequent detection of the same intrusion having similar behavior.
Attack State	A state where we conclude that some intruder is purposely trying to gain unauthorized access to single node or an entire network
Collector	A high-level node that receives alerts and events from all detection sensors
DHT	A distributed hash table, containing key value pair of the event and collector information

proposed a generic framework to introduce blockchain and distributed ledger-based technologies to IDS for trust management between collaborating nodes. NSTAT uses a similar centralized approach for analysis but its operations are in real-time. Moreover, in NSTAT the alerts generated by detection sensors are filtered and pre-processed before sending for analysis [12]. Centralized server analyzes events against pre-defined scenarios and try to correlate them by using state transition mechanism. Hamed *et al.* [13] surveys a number of techniques to pre-process data for IDS. A number of similar studies has been conducted in the recent past to collectively analyze the advancements in the field of alert fusion. One such study was conducted by Ramaki *et al.* [14] where the authors reviewed hundreds of recently proposed solutions. Statistical methods are being used to detect events and discover previously unknown patterns using data mining approaches [15], [16].

Many issues of the centralized approach like scalability and single point of failure can be solved by hierarchical structures. Different approaches for hierarchical structures are proposed in the literature. The Indra system is most effective against scan attacks, e.g., Port Scan attack, in which same vulnerability is exploited for several nodes at a single time. Whenever the nodes detect some malicious activity, it informs the neighboring nodes, so that they can enhance their defenses for some specific attack rather than adapting some generic security procedure [17].

There are several DIDS implementations that adopt publish/subscribe models, including [18] and [19]. The basic idea behind publish/subscribe is that using the application level multi-cast lets user's login into different groups of interest and, within the group, users can share information to thwart different attacks. Publish/Subscribe modes work well for small and moderate scale IDS but ultimately it suffers from $O\left(n_i^2\right)$ communication. In the worst case if each member of a particular group is under attack, then every member of the group will send messages to every other member. This is not practical in today's global scale attacks as every group can have potentially thousands or even millions of subscribers. Different from the above solutions, our original proposal sends messages to only handful of collectors and every node receives the fusion report from its collector and alter detection parameters accordingly. Thus, in our solution, as better detailed in the following, communication between nodes and collectors only suffers from $2 \times n_i$.

Structured networks in a peer-to-peer environment generally implements DHT. DHT works like a hash map, and information can be stored in a DHT dimension in the form of a (key, value) pair. The data can be reliably retrieved if the key is known precisely. Distributed Hash Table provides lookup service for information in Peer to Peer (P2P) applications. Nodes are distributed uniformly across a key space and all participating nodes form an overlay network. Each node maintains a routing table containing information about neighboring nodes.

DHT is decoupled from the physical network topology and DHT-based applications use middleware for simplification of development and transparency. There are several DHT implementations available, each different from the other in terms of lookup, distance calculation, and storing routing information. Kademlia uses tree-based routing differently from Chord, CAN, or Pastry. Node identifiers, files or anything that needs to be located by the Kademlia DHT lookup function is deployed using SHA-1 hash into 160 bits space. Like Chord every node keeps more information of its neighbors and less about far nodes. The distance between two nodes is measured as bitwise XOR between nodes.

$$distance\,(a, b) = a \oplus b$$

The DARPA [20] dataset is among the famous datasets for intrusion detection evaluation. It has been used by many researchers including [21]; it is a safe and logical choice for most of the researchers to test new detection systems. This dataset can be downloaded from [22] and it is free to use. DARPA 1999 dataset consists of many different connections, both malicious and benign and each connection has 41 features. DARPA KDD99 datasets consist of 39 different types of attacks divided in four categories. 1) DDoS, 2) Probe, 3) User to Root, and 4) Remote to Local. Many duplicate connections are removed from time to time and the text labels have also been replaced with numeric values. The values of features have also been normalized to give equal importance to each feature. Despite some criticism DARPA 1999 dataset is still one of the most popular amongst researchers throughout the world as it is the pioneer of providing the dataset to the research community. Both datasets provided by DARPA are offline and all changes suggested in 1998 dataset are addressed in 1999 dataset.

III. RELATED WORK

Data fusion is the process involving the input stream of multiple nodes for detection, aggregation, estimation, and combination to find a semantic relationship between data in order to achieve higher level of abstraction for well informed decision making [23]. Information collected from multiple sources

for the purpose of decision making often leads to more reliable results. Snort [24] is widely used as a network-based intrusion detection and it is a de-facto standard among other intrusion detection systems. It is widely acceptable in the research community and is highly sophisticated in terms of usability and functionality. Snort is signature-based intrusion detection system, meaning, it sniffs the network traffic and detects network traffic anomalies by comparing it with predefined set of rules. It can also generate log files and examine the content of each individual packets. Snort is divided into four logical modules, 1) Packet Decoding 2) Preprocessor 3) Detection Engine, and 4) Alert and Logging. Snort can also be used as real-time event detection and is usually robust. We have compared all our results with events generated by Snort sensors.

There is a number of intrusion alert fusion frameworks in the recent literature. However, alert fusion and correlation in the context of IoT is lacking. Only a few recent implementations are proposed and most of them are still in the conceptual design stage. Our proposed framework exploits the IoT infrastructure to propose a solution deployed on the edge of the network to reduce the amount of computations required. The fog computing has a unique advantage of providing computations over network edges, thus not only reducing hosting costs but also decreasing the bandwidth and data flow between nodes. Modern IDSs use multiple sensors for intrusion detection. The use of multiple sensors is more reliable and it maximizes the efficiency of the overall system thereby reducing the false alarm rate. Multi sensor data fusion or Distributed Fusion [25], combine information from multiple sources and various other output sources from the network. Combining multiple pieces of information for decision about a certain situation or event is more informed and hence more reliable. Collectors can provide details about the intruder, its location, duration of attack, severity of the observed threat, the frequency of attack on the entire network, any specific process or service being attacked etc. Below we discuss several techniques used in the literature for alert management.

After the deployment of any IDS, the alert management is usually the next step. Typical IDS not only detects malicious attempts to break into the system but also deals with large number of unwanted alerts that are generated constantly. This is particularly true in DIDSs as participating peers can join and leave at any time. Reducing these large amounts of intrusions is known as alert management or distributed alert management for distributed systems. There are many ways to manage alerts generated by IDS. Low level alert management is where each attack is treated and ranked individually on the basis of its importance. This process usually prioritizes alerts on the basis of its potential risks, the assets that are being attacked, the attacker that is initiating the attack etc., [26]. High level alert management is different from low level alert management because it does not look into each and every alert, instead it uses clustering, fusion, correlation, and aggregation processes to form an abstraction of alert generated and ranked by low level management. High level management often improve the overall outcome of the IDS [27]. The accuracy and efficiency of IDS is improved mainly because of the evaluation being done in the low-level management. Almost every alert management technique has to deal with the large number alerts being generated by different heterogeneous or homogeneous sensors.

Alert reduction is a process of reducing alerts generated by multiple detectors. This can be implemented in both detection phase as well as in the fusion process of the alert. Alerts can be reduced by using a number of different techniques, some of which include, aggregation, merging, and clustering etc. Meng and Kwok [28] proposed a machine learning based approach to reduce false alarms in Network IDS by use of ensemble classifiers. Alert aggregation is where we combine multiple alerts into a group on the basis of some common parameters such as same type of attack, same source IP etc., [29]. In the merging process we combine similar alerts into one single alert, making it simple to read, transport, and it presents an abstract view of the whole event. Modern attackers use sophisticated attacks which is often a series of events. To thwart such complex attacks, event correlation plays an important role. With alert correlation one can try to find a link between certain types of events and make a scenario out of these otherwise random events [30]. Without correlation false positive rate is very high and it makes the process of analysis much harder. Alert correlation not only helps reduce the false positive rate but also reduces the large number of messages that may overload the system [31]. Usually any IDS produces alerts whenever it encounters some malicious activity. This single alert is composed of many different parameters including Target IP, Source IP, Source Port, Target Port, Timestamp, Attack Type etc. Presenting all these parameters in a convenient and user-friendly way is the key to early detection [32]. De Alvarenga *et al.* [33] proposed a high-level visualization model using process mining for network administrators to analyze and cluster complex scenarios intuitively. The events collected from different sensors are usually heterogeneous and require normalization before feeding them to correlation modules. The goal of this process is to encode and translate all the properties of an alert and format them in a commonly understandable format. To achieve this, [34] proposed a data model to encode different formats of the alert to be represented in a common one. The main theme behind Intrusion Detection Message Exchange Format (IDMEF) is to provide semantics for each attribute of the alert detected. It is highly customizable and is left up to the implementation to define names for each field; this results in a very flexible solution that each collector defines alerts based on their understanding and availability.

In our case, the framework implements the normalization by using a wrapper module that collects and translates different attributes of each alert from sensors and converts them using a knowledge base presented in Table 2. The names of these alerts are collected from [32] wherever possible.

TABLE 2. Alert normalization with attributes and description.

Attribute	Description
ID	A unique identification of each alert
Time	The time at which the event was detected
Name	Name of the alert that was detected by IDS
Priority	The discrete priority that was assigned
Sensor	The ID of the sensor that detected the alert
Start Time	The start time of the attack
End Time	The end time of the attack
Collector	The ID of the collector that the alert was fused
Target Process	The name of the process that was attacked
Target Port	The name of the port that was attacked

Common Vulnerabilities Exposures is a list for all common alerts with all the standard names of the alerts. However, not all names of alerts and fields are readily available; in the case of need, we assigned our own names i.e., Alert 1, Alert 2 etc.,

Each alert is translated and assigned a name along with its description before forwarding. This helps normalize all the alerts and reduce the heterogeneity of the system.

Alert Prioritization is another technique of managing huge number of alerts generated by detection sensors. Alerts are prioritized in a number of different ways and important alerts are distinguished from the less important ones. Assigning priority to alerts helps reduce the number of messages generated and it also helps consume less network resources. There are different Prioritization schemes available. For instance, [35] uses a fuzzy-logic based scheme for alert scoring and rescoring.

IV. PROPOSED FRAMEWORK

Here we present an original framework, which has the primary goal of achieving better scalability and performance results by minimizing the number of alerts generated by IDS sensors in IoT based edge network. The proposed architecture works in two primary phases. In the first phase, events generated by IoT sensors are routed through fog nodes and then each event is normalized, pre-processed, and filtered to remove less important events. Then, we evaluate different metrics that assign priorities to the pre-processed events. In the second phase, we discuss the routing mechanism by which these alerts are disseminated to higher nodes in our hierarchical architecture for further analysis and decision making. Higher level nodes collect and correlate pieces of information passed on to them and raise alarms if necessary. Figure 1 shows the proposed framework.

As discussed earlier in Section II, messages generated by sensors are passed on to their respective collectors using DHT for analysis. The Kademlia implementation of the DHT is used to store and route alert information. Collectors use correlation and clustering techniques to find anomalies and raise alarms if necessary. Collectors also inform neighboring nodes to increase priority for certain attacks, ports, services, applications, or black listed IP addresses. Nodes of the

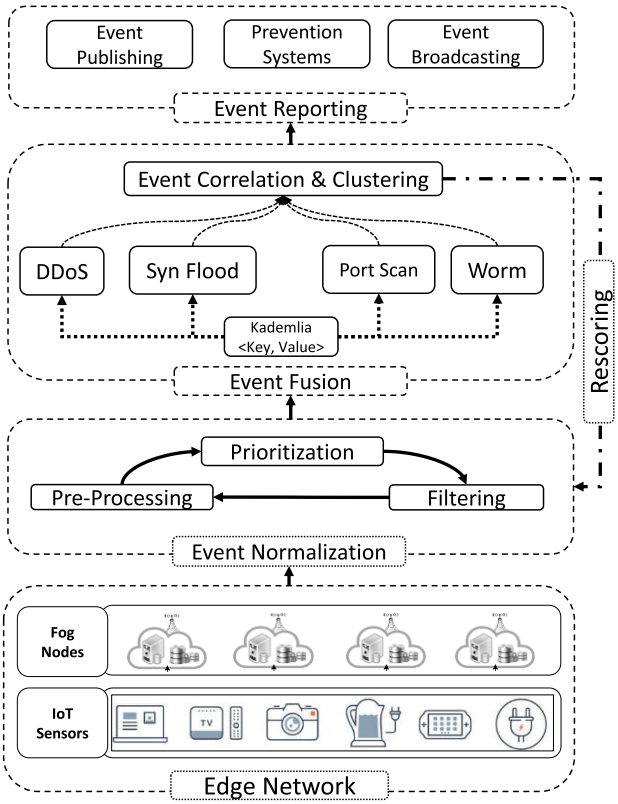

FIGURE 1. Our proposed framework for intrusion alert fusion in IoT based IDS sensors.

Kademlia DHT form an overlay network. Each node maintains a routing table to find nodes and consistently querying nodes for communication and maintenance of network. Every node keeps more information about their neighbors and less about farther nodes. Information retrieval and querying in a scalable distributed network is resource intensive and often not secure. DHT-based solutions provide more facilities than any other distributed architecture as shown in Table 3. DHT is more reliable, fault-tolerant, scalable, and easy/transparent to use. The overlay network it establishes has no single-point-of-failure and if the nodes of DHT overlay are under attack, the computational load as well as the network load are uniformly balanced among nodes, keeping network healthy and

TABLE 3. DHT comparison with other alternatives [38].

Facility	Abstraction	Ease of Use	Scalability	Load Balancing
DHT	High	High	High	Yes
Centralized Lookup	Medium	Medium	Low	No
P2P Flooding Queries	Medium	High	Low	No
Distributed Filesystem	Low	Medium	Medium	No

reliable. We used Kademlia DHT for routing of information to respective collectors because it suits best for situations where number of messages are very high.

The number of message generation for Chord is $log_2 N$ for every event, where N is the total number of nodes participating in the overlay. Moreover, for sending alert information from one node to the other Chord requires $\left(n \cdot \log_2 N\right)/n$ while Kademlia requires only $\left(n + \log_2 N\right)/n$ making it more suitable in the situation of distributed sensors. Table 4 gives a clear overview of both technologies.

TABLE 4. Analysis of cost in structured overlay.

Overlay	Chord	Kademlia
Routing Algorithm	Recursive	Iterative
Node Lookup	0	$\log_2 N$
First Event Stored	$\log_2 N$	$1 + \log_2 N$
n events stored in the same key	$n \cdot log_2 N$	$1 + \log_2 N$
Average number of messages per event	$(n \cdot log_2 N)/n$	$(n + log_2 N)/n$
Average number of messages when $n \to \infty$	$\log_2 N$	1

The frequency of attacks over the years have increased tremendously, and any global scale attack is registered by many IDSs throughout the world, which can easily overwhelm some of the collectors. Design of the alert key is very important to route similar alerts to same collectors without affecting the load distribution on the overlay network. For this purpose, we use only abstract level characteristics of the event to spread these events more evenly to all collectors. Since the number of collectors in the overlay are far less than those of IDS and nodes, we introduce a new key design technique to classify all events to their respective collectors evenly and get a global view of all related events. In order to present the significance of this technique we take IP/Port Scanning attack as an example, since this is the most common type of event on the Internet.

Previous studies suggest that scanning attacks are the most common and leave a significant distribution tail behind i.e., a large portion of all scan attacks are usually initiated by very small number of nodes. Moreover, the popularity of each port and IP address is not consistent even within hours of the day [36]. Since the alert information is being passed as part of the key to the collectors, we need to incorporate only the prefix of the IP address in order to keep the size of the key small. Since the DARPA 1999 dataset lacks the sheer number of scan events required for the experiment, we analyzed the stability of the collectors by the dataset obtained from DShield [37]. This data consists of one month of network logs which is more than 600 million in total and it is shared by more than 2000 volunteers throughout the globe.

We look into the stability of collectors during overwhelming number of messages generated by sensors. It is observed that after sufficient amount of time, large percentage of port scan attacks are usually from 64 popular ports. DShield data suggests that more than 85% of port scan attacks are from these popular ports only, and this continues for up to three weeks. On the other hand, the stability of IP Address is not consistent. The popularity of class C IP address (/24) is not stable mainly because there are no steady blacklists of IP addresses. Ports scan analysis tend to be more stable than IP address and hence more reliable. Leveraging this, we use only 7 bits of Kademlia DHT key in a 127-bit bucket. All the popular ports are routed to their respective collectors and the rest is distributed randomly to the remaining collectors. Using this technique, we effectively achieve load balancing. This is more effective in situations where the number of collectors is relatively low.

Alert information generated by heterogeneous sensors can be different from each other and summarizing them into one unified format is necessary for efficient evaluation. Alert representation of each sensor is different and depends upon the network they are deployed on and the assets they are trying to defend. In most cases the information being shared among different nodes and IDSs are either incomplete or in the format that is not usable, for instance the date format and IP for UNIX based systems are represented in numeric format, this is also true in situations where same event is named differently by different sensors. Figure 2 shows a snippet of the log file that is labeled after converting it to IDMEF [38]. We can see that it assigned attack names to events detected by sensors along with all the other details like Source IP, Dest Port, Duration etc., In this module each alert is formatted into a common, more usable format. This format consists of fields (parameters) that are common among multiple sensors. The reason to represent alert information into a common format is because the Correlation and Fusion Module needs to understand the alert information more easily in otherwise heterogeneous environment. In most cases for common alert representation, IDMEF is used as the output of the sensors.

```
<?xml version="1.0" encoding="UTF-8"?>

  <idmef:IDMEF-Message version="1.0"
                       xmlns:idmef="http://iana.org/idmef">
    <idmef:Heartbeat messageid="abc123456789">
      <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
        <idmef:Node category="dns">
          <idmef:location>Headquarters DMZ Network</idmef:location>
          <idmef:name>analyzer01.example.com</idmef:name>
        </idmef:Node>
      </idmef:Analyzer>
      <idmef:CreateTime ntpstamp="0xbc722ebe.0x00000000">
        2000-03-09T14:07:58Z
      </idmef:CreateTime>
      <idmef:AdditionalData type="real" meaning="%memused">
        <idmef:real>62.5</idmef:real>
      </idmef:AdditionalData>
      <idmef:AdditionalData type="real" meaning="%diskused">
        <idmef:real>87.1</idmef:real>
        <severity:real>3.5643</severity:real>
      </idmef:AdditionalData>
    </idmef:Heartbeat>
  </idmef:IDMEF-Message>
```

FIGURE 2. Our IDMEF-compliant log file.

The final output of the sensor is stored in local databases for future use.

Alert filtering module is necessary for scraping unnecessary events before fusing them to collectors Alert filtering process can affect the overall performance of the system thereby providing accurate results and have significantly less false alarms. We filter events based on Algorithm 1 presented in Table 5. We first generate the key from the event detected, then search the local database for similar key, if the key is found once and only once, then this means that the event is isolated and it has no significant value to be reported. The Alert Filtering algorithm can have limited impact on the overall performance especially where the nature of attack is more complex and sophisticated. We extract the necessary information of the event and check it in the global events relational database and check for its occurrence in any global scale attack. Using this technique, we can filter out many less important events while maintaining the registration of high priority events.

TABLE 5. Alert filtering by removing isolated events.

Algorithm I

procedure CORRELATION
  Require: Logfile **l**
  Input: Event *e*
  while *Processing* **do**
    if detected **then**
      key ⇜ *GetEventKey*()
      count ⇜ GetEventCount(*key*)
      if *count* == *1* **then**
        Label as *FP*
      else
        Label as *TP*

To reduce unnecessary messages several techniques are proposed in the literature, some of which are discussed in Section III. Prioritizing intrusion events is necessary because all alerts are not equally important. Different parameters of each alert are analyzed and score is assigned to each parameter in every step. The weighted average of these scores results in a commutative value (cv), this value is compared to a threshold (tv), and event is fused to its respective collector if this threshold is crossed i.e., *if* $cv > tv \rightsquigarrow fuse$. To calculate cumulative value for each event, different parameters are analyzed, each resulting in a factor used in our original overall metric. These parameters are discussed below.

1) TARGET

This metric is used to evaluate the target machine, service, application or port number being attacked by the adversary. This metric outputs a higher value for more valuable asset and results in lower values if it is less important. Equation 1 is used to calculate the importance of the node being targeted in the attack.

$$I(t) = \Pi w(a) \tag{1}$$

Here "a" is any important services, file, directory, subnet, and port etc., for a target machine. Important assets are identified by administrators manually.

2) EVENT SEVERITY

Severity of any event is the threat that it poses to the overall security of the network. This value is calculated by several sources on the internet, including Common Vulnerabilities and Exposures CVE [39], LSVA [40] etc., We have used these values in the overall scores of the cumulative value presented in Equation 2. But these severity scores provided by these companies vary, each has its own criteria of assigning values to vulnerabilities. Expected severity can be calculated by the following equation:

$$S(e) = \frac{\sum_{i=1}^{n} w(eS_i) \times eS_i(a)}{\sum_{i=1}^{n} w(eS_i)} \tag{2}$$

3) DETECTION CONFIDENCE

The detection confidence matrix is calculated using the Equation 3. This metric is used to check if the detection algorithm by which the actual detection is made is trust worthy. Past experience has shown that anomaly-based detectors are less effective than signature based.

Conf(a) gives the computed value of detection algorithm a, where a can be signature-based, anomaly-based or any other type. $\overline{E_{a,t}}$ gives the average value for algorithm a in time t for list of efficient algorithms E or Less Efficient (LE).

$$\begin{aligned} Conf(a) &= \overline{\mathrm{E_{a,t}}} \times \mathrm{ES(event)} \quad \text{wheni} \in \mathrm{E_i} \\ Conf(a) &= \overline{LE_{a,t}} \times ES(event) \quad when\ i \in LE_i \end{aligned} \tag{3}$$

4) ALERT CORRELATION

In a large-scale collaborative attack, the attacker uses a sequence of steps and each step brings him closer to the desired goal. The priority of an alert will be high if the correlation value of an event is high. This correlation can be between source and target IP, services and ports being attacked etc. The similarity of the two parameters is calculated in Equation 4.

$$C_{param(a1,a2,\ldots an)} = \frac{\sum similarbits(a1, a2, \ldots an)}{\sum_{t}^{c-\Delta} allbits(a1, a2, \ldots an)} \tag{4}$$

In Table 6 we present an algorithm to calculate the correlation value of the event e if it finds the event in the logfile l

TABLE 6. Algorithm II correlating similar events.

Algorithm II

procedure CORRELATION
  Require: Logfile **l**
  Input: Event *e*
  K = {$\boldsymbol{e} \mid \boldsymbol{e} \in \mathbf{l} \wedge \boldsymbol{time(e)\ in\ \Delta}$}
  while |K| *is_not_empty* **do**
    if $\boldsymbol{e} \in$ K **then**
    value ← $\mathbf{C_{param}(e)}$
    max ← $\mathbf{C_{param}(e)}$
      if *value* > *max* **then**
        max ← $\mathbf{C_{param}(e)}$

and the time of event is closer to time $\boldsymbol{\Delta}$, here $\boldsymbol{\Delta} = 12$ hours. The algorithm will compare the correlation value of this event with already stored maximum value of similar event. If this value is higher than the overall correlation value, it will be reported as high. We have only considered the values between Source IP, Dest IP, Source Port, and Desk Port, but it can be calculated for other parameters.

5) SERVICE HISTORY

Historically some alerts are more common than others and some vulnerabilities are exploited more often than others. CVE [32] provide information about the stability of a certain service in the recent history. Equation 5 calculates historical severity of a specific service by weighted arithmetic mean.

$$\boldsymbol{H}_{(s)} = \frac{\sum_{e \in V_h(s)} \boldsymbol{\omega}(s) \times \boldsymbol{S}(e) \times \boldsymbol{\gamma}^{-\boldsymbol{\Delta}(sv)}}{\sum_{e \in V_h(s)} \boldsymbol{\omega}(s)} \tag{5}$$

Old service history is given lower value as compared to more recent ones and very old are ignored altogether as shown by $\gamma^{-\Delta(sv)}$. The event severity of service s is considered while calculating historical vulnerability.

$$cv(e) = \eta_1 \times I(t) + \eta_2 \times S(e) + \eta_3 \times Conf(a) + \eta_4 \times C_{param}(e) + \eta_5 \times H_{(s)}$$

Finally, the overall cumulative value of event e is calculated as weighted average of all the other metrics. In our approach, the alert scores are generated for each event, and seriousness of the alarm is calculated by the above described metrics. Each output from each metric is aggregated and a single numeric value is generated on the basis of which we decide the severity of the event and only those events are passed on to the collectors for correlation. The technique is compared with Snort and results are presented in the next section. Table 3 summarizes the events detected by Snort and the corresponding priority assigned to each event. The true ICMP Echo Request is registered by Snort 786 times while our priority metrics only prioritized 84. Similarly, the RPC Portmap Sadmind request UDP is registered 290 times by Snort, while it is prioritized by our system only 88 times assigning the score of 7.7. The score assigned to True events are relatively high thereby registering them to collectors, while False Events are kept between low scores of 1 and 2.2 and are not sent to collectors. These events are marked as low-level events and hence they are ignored by the local IDS, these events will eventually be sent to its respective collector if it crosses the threshold in time.

V. RESULTS

In this section we discuss results of the proposed system. Each intrusion event has to be assigned a key, in order to insert it to a DHT overlay. Selection of key for each event is the IP address of the adversary found in almost all security log files, the same key was used for correlation purpose. Each IDS sensor sends intrusion events to collectors after a certain threshold is crossed, this is achieved by assigning severity score to each alert. This helps us evaluate the performance of our system.

Nodes in the overlay network use heterogeneous sensors. Alerts generated by each involved sensor are different from each other. They are made uniform by using IDMEF library [38] before fusing them to their respective collectors. Collectors deployed for intrusion alert collection are assumed to be static and does not change. They are selected on the basis of their security and available bandwidth, however, there can be multiple parameters for selecting and re-selecting collectors dynamically to adapt to changing conditions. Selection and dynamic adjustment of collectors in the DHT overlay network is out of the scope of this specific work. Our proposed framework is tested against the DARPA 1999 dataset. DARPA presented numerous datasets for training and evaluation of similar systems. DARPA intrusion detection datasets include 1998, 1999, and 2000 datasets. All alerts generated from the DARPA 1999 dataset are stored in a local database, MySQL in our implementation case [41]. We are aware that MySQL database is not the optimal choice for our framework (see future work directions) but was selected to rapidly provide a proof-of-concept based on very widespread and adopted technology. Data generated by sensors are stored locally in their own database table and collectors' data is stored in a separate table. The proposed algorithms are implemented in Java and the library for Kademlia DHT implementation is used.

Figure *3* shows the total number of messages generated by sensors with and without using prioritization schemes. The graph shows total number of messages generated before and after applying prioritization scheme.

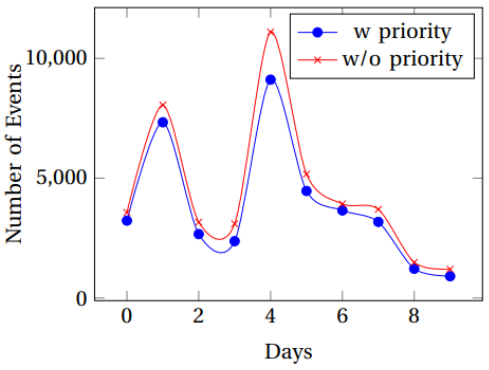

FIGURE 3. Alert reduction over 10 days of data.

It is clear from Figure *4* and Figure *5* that our proposed method significantly reduces less important alerts. The reported results also show that we are able to reduce the alerts more for larger numbers, while for small numbers of original messages produced by Snort sensors, they are comparatively less reduced.

This is because the prioritization scheme used here reduces the threshold value relatively quickly for small number of alerts than for large number to avoid exaggeration attack [42].

Figure 6 shows the alert scoring technique and its effectiveness in identifying false positives. For instance, Snort detected an event and label it as an 'overflow' intern

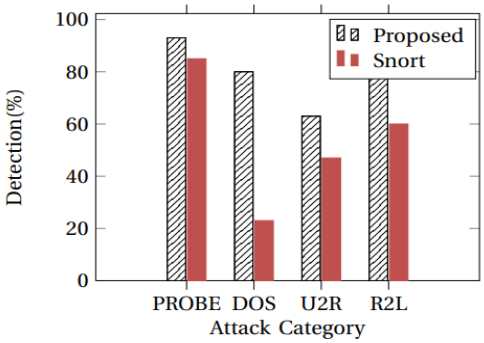

FIGURE 4. Attack wise reduction rate–our proposal vs Snort.

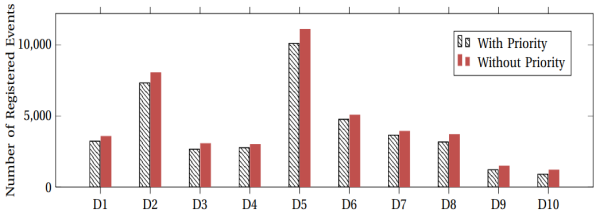

FIGURE 5. Day wise attack reduction rate.

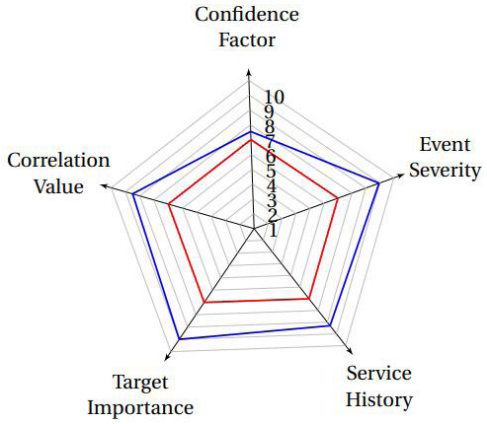

FIGURE 6. Alert metrics evaluation–our proposal vs Snort.

assigning it the priority of 6. On the other hand, our approach labeled it as low-level event because it is impossible to occur given the applicable context (i.e., destination address running on Mac operating system). As we can see, Snort labels almost all alerts as medium while our approach manages to distinctly and successfully separate higher level alerts and lower level alerts. This approach also identifies the series of events that may be part of a large-scale attack against the network. The value is calculated on the basis of the metrics discussed in Section IV.

Table 7 presents the summary of the generated alerts. They are split into two categories, i.e., True Positives and False Positives. In the experiments, our solution not only identified the alerts correctly, but also assigned adequate priority based on the severity of the associated attack. In addition, it performed better than Snort not only in terms of detection accuracy but also by reducing alerts to more than 62%.

TABLE 7. Summary of the prioritized events.

Event	Snort	Proposed	CV
True Positive			
ICMP Echo Request	768	84	≈ 1.7
ICMP Echo Reply	39	6	≈ 1.67
RPC Portmap Sadmind request UDP	290	88	≈ 7.6
RPC Sadmind UDP PING	9	6	≈ 5.3
RPC Sadmind with root attempt UDP	44	27	≈ 9.1
BAD-TRAFFIC loopback traffic	144	144	≈ 8.8
False Positive			
NETBIOS NT NULL session	1	0	1
ATTACK-RESPONSES Invalid URL	1	0	1
SNMP request UDP	8	1	2.2
SNMP public access UDP	7	1	2.1
ATTACK-RESPONSES 403 Forbidden	2	0	1
MS-SQL version overflow attempt	1	0	1
ICMP redirect host	24	1	1

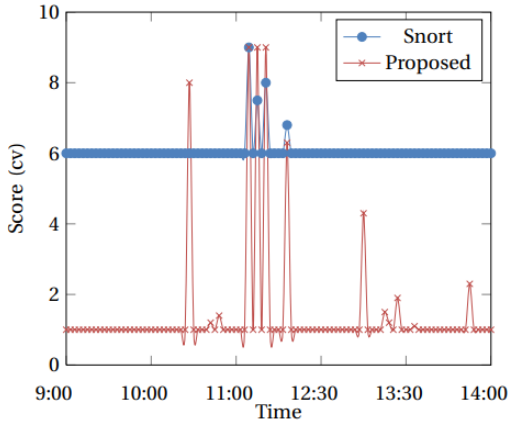

FIGURE 7. Alert prioritization score.

Figure 7 shows another relevant aspect of our approach: unlike Snort, it assigns priority levels varying from medium to high (cv = 1 to 10). Let us remind that Snort only assigns priority to events where the IP address of the adversary is in the blacklist, leaving all others as medium.

Alert information of each event is shared by multiple detection sensors. It is more likely that one event might be chosen by tens and thousands of detection sensors. All this information has to be processed and stored in the overlay network of the DHT. The overlay network can easily process this information as it can disseminate the alert information to each individual collector, thus reducing the overall load. Figure 8 shows the accuracy of the prediction rate. Over the course of 10 days the system maintained the accuracy rate above 80%, again while reducing messages to 62%.

After extensive experiments, on different datasets, we came to the conclusion that the proposed technique can be used to prioritize most of the datasets available today. However, statistical approach to prioritize might not be appropriate for larger datasets as this is expensive approach and is

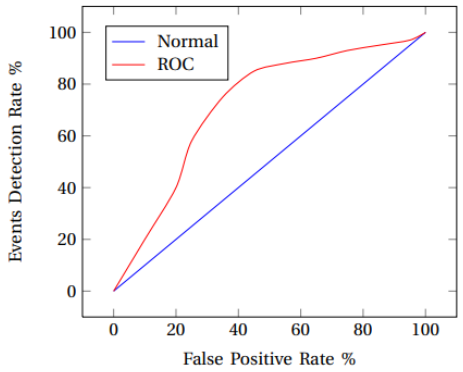

FIGURE 8. ROC curve of the false positive and detection rate.

very difficult to find meaningful patterns in larger datasets. The technique works for peer-to-peer networks and extensive simulations were conducted to get desired results. Further testing in different non-simulation-based environment might benefit for further fine tuning of the system.

VI. CONCLUSION

Attacks on large-scale networks are growing rapidly and thwarting these attacks is highly sophisticated and extremely difficult to handle. Pieces of attack information are spread across multiple nodes and combining and analyzing them in an intelligent, distributed, and scalable way poses several technical challenges. In this article, we proposed a system that is capable of relevantly reducing the number messages generated by IDS deployed across the Internet in a distributed fashion. Our solution can analyze an intrusion event and assign priority to each alert based on diverse metrics and attributes. Then, after events are associated with adequate priorities, they are forwarded to a Kademlia-based DHT overlay, which is inserted in the middle (as a mediator) between detection sensors and collectors. Collectors gather intrusion information from different sources and analyze them by using correlation functions. Identical attacks are routed to a same collector by assigning same keys for similar attacks. This mechanism largely reduces the number of messages generated by detection sensors while reducing at the same time the false alarm rate.

Given the encouraging results achieved so far, we have already started significant research work to extend the proposal originally presented here. First, we are planning to validate and evaluate the proposed solution over other datasets available in the community. Second, we are working to replace MySQL with a non-structured persistent storage such as MongoDB. Finally, we are exploring some additional priority metrics that can be helpful to assign priority levels to alerts in a more practical way.

REFERENCES

[1] SophosLabs Research Team, "Emotet exposed: Looking inside highly destructive malware," *Netw. Secur.*, vol. 2019, no. 6, pp. 6–11, 2019.

[2] Security Magazine. (2019). *First Three Quarters of 2019: 7.2 Billion Malware Attacks, 151.9 Million Ransomware Attacks*. Accessed: Jun. 28, 2020. [Online]. Available: https://www.securitymagazine.com/

[3] W. Li and W. Meng, "Design of intrusion sensitivity-based trust management model for collaborative intrusion detection networks," in *Proc. IFIP Int. Conf. Trust Manage.* Berlin, Germany: Springer, 2014, pp. 61–76.

[4] Y. Meng, "Adaptive false alarm filter using machine learning in intrusion detection," in *Practical Applications of Intelligent Systems*. Berlin, Germany: Springer, 2011, pp. 573–584.

[5] T. Hussain, K. Muhammad, S. Khan, A. Ullah, M. Y. Lee, and S. W. Baik, "Intelligent baby behavior monitoring using embedded vision in IoT for smart healthcare centers," *J. Artif. Intell. Syst.*, vol. 1, no. 15, p. 2019, 2019.

[6] W. Li and W. Meng, "Enhancing collaborative intrusion detection networks using intrusion sensitivity in detecting pollution attacks," *Inf. Comput. Secur.*, vol. 24, no. 3, pp. 265–276, Jul. 2016.

[7] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. Workshop Econ. Peer-Peer Syst.*, vol. 6, 2003, pp. 68–72.

[8] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth multicast in cooperative environments," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 298–313, Dec. 2003.

[9] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *Proc. Int. Workshop Netw. Group Commun.* Berlin, Germany: Springer, 2001, pp. 14–29.

[10] M. Saikia, N. Hoque, and D. K. Bhattacharyya, "MaNaDAC: An effective alert correlation method," in *Recent Developments in Machine Learning and Data Analytics*. Singapore: Springer, 2019, pp. 249–260.

[11] N. Alexopoulos, E. Vasilomanolakis, N. R. Ivánkó, and M. Mühlhäuser, "Towards blockchain-based collaborative intrusion detection systems," in *Proc. Int. Conf. Crit. Inf. Infrastruct. Secur.* Cham, Switzerland: Springer, 2017, pp. 107–118.

[12] R. A. Kemmerer, "NSTAT: A model-based real-time network intrusion detection system," Dept. Comput. Sci., Univ. California, Santa Barbara, Santa Barbara, CA, USA, Tech. Rep. TRCS97-18, 1997. [Online]. Available: http://www.cs.ucsb.edu/TRs/TRCS97-18.html

[13] T. Hamed, J. B. Ernst, and S. C. Kremer, "A survey and taxonomy on data and pre-processing techniques of intrusion detection systems," in *Computer and Network Security Essentials*. Cham, Switzerland: Springer, 2018, pp. 113–134.

[14] A. A. Ramaki, A. Rasoolzadegan, and A. G. Bafghi, "A systematic mapping study on intrusion alert analysis in intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 1–41, 2018.

[15] A. Sultana and M. A. Jabbar, "Intelligent network intrusion detection system using data mining techniques," in *Proc. 2nd Int. Conf. Appl. Theor. Comput. Commun. Technol. (iCATccT)*, 2016, pp. 329–333.

[16] E. Ariafar and R. Kiani, "Intrusion detection system using an optimized framework based on datamining techniques," in *Proc. IEEE 4th Int. Conf. Knowl.-Based Eng. Innov. (KBEI)*, Dec. 2017, pp. 0785–0791.

[17] R. Janakiraman, M. Waldvogel, and Q. Zhang, "Indra: A peer-to-peer approach to network intrusion detection and prevention," in *Proc. 12th IEEE Int. Workshops Enabling Technol., Infrastruct. Collaborative Enterprises (WET ICE)*, 2003, pp. 226–231.

[18] T. Shekari, C. Bayens, M. Cohen, L. Graber, and R. Beyah, "RFDIDS: Radio frequency-based distributed intrusion detection system for the power grid," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–15.

[19] A. V. Uzunov, "A survey of security solutions for distributed publish/subscribe systems," *Comput. Secur.*, vol. 61, pp. 94–129, Aug. 2016.

[20] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Comput. Netw.*, vol. 34, no. 4, pp. 579–595, Oct. 2000.

[21] A. M. Lonea, D. E. Popescu, and H. Tianfield, "Detecting DDoS attacks in cloud computing environment," *Int. J. Comput. Commun. Control*, vol. 8, no. 1, pp. 70–78, 2013.

[22] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.

[23] C. Siaterlis and V. Maglaris, "One step ahead to multisensor data fusion for DDoS detection," *J. Comput. Secur.*, vol. 13, no. 5, pp. 779–806, Dec. 2005.

[24] M. Roesch, "Snort: Lightweight intrusion detection for networks," *Lisa*, vol. 99, no. 1, pp. 229–238, 1999.

[25] M. A. Bakr and S. Lee, "Distributed multisensor data fusion under unknown correlation and data inconsistency," *Sensors*, vol. 17, no. 11, p. 2472, Oct. 2017.

[26] S. Hiruta, S. Ikeda, S. Shima, and H. Takakura, "IDS alert priority determination based on traffic behavior," in *Proc. Int. Workshop Secur.* Cham, Switzerland: Springer, 2019, pp. 189–206.
[27] M. Wu and Y. Moon, "Alert correlation for cyber-manufacturing intrusion detection," *Procedia Manuf.*, vol. 34, pp. 820–831, Jan. 2019.
[28] Y. Meng and L.-F. Kwok, "Enhancing false alarm reduction using voted ensemble selection in intrusion detection," *Int. J. Comput. Intell. Syst.*, vol. 6, no. 4, pp. 626–638, Aug. 2013.
[29] S. Saad and I. Traore, "Extracting attack scenarios using intrusion semantics," in *Proc. Int. Symp. Found. Pract. Secur.* Berlin, Germany: Springer, 2012, pp. 278–292.
[30] H. T. Elshoush and I. M. Osman, "Intrusion alert correlation framework: An innovative approach," in *IAENG Transactions on Engineering Technologies*. Dordrecht, The Netherlands: Springer, 2013, pp. 405–420.
[31] V. Heorhiadi, M. K. Reiter, and V. Sekar, "New opportunities for load balancing in network-wide intrusion detection systems," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2012, pp. 361–372.
[32] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," in *Proc. 1st-Forum Incident Response Secur. Teams*, vol. 1, 2007, p. 23.
[33] S. C. de Alvarenga, S. Barbon, Jr., R. S. Miani, M. Cukier, and B. B. Zarpelão, "Process mining and hierarchical clustering to help intrusion alert visualization," *Comput. Secur.*, vol. 73, pp. 474–491, Mar. 2018.
[34] H. Debar, D. Curry, and B. J. Feinstein, *The Intrusion Detection Message Exchange Format (IDMEF)*, document RFC 4765, 2007.
[35] F. Saidi, Z. Trabelsi, and H. B. Ghazela, "Fuzzy logic based intrusion detection system as a service for malicious port scanning traffic detection," in *Proc. IEEE/ACS 16th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2019, pp. 1–9.
[36] V. Yegneswaran, P. Barford, and S. Jha, "Global intrusion detection in the DOMINO overlay system," Dept. Comput. Sci., Univ. Wisconsin-Madison, Madison, WI, USA, 2003.
[37] H. Koike, K. Ohno, and K. Koizumi, "Visualizing cyber attacks using IP matrix," in *Proc. IEEE Workshop Vis. Comput. Secur. (VizSEC)*, 2005, pp. 91–98.
[38] H. Debar, D. Curry, and B. Feinstein, *The Intrusion Detection Message Exchange Format (IDMEF)*, document RFC 4765, 2007.
[39] M. Panda, A. Abraham, and M. R. Patra, "A hybrid intelligent approach for network intrusion detection," *Procedia Eng.*, vol. 30, pp. 1–9, Jan. 2012.
[40] V. Vlachos and D. Spinellis, "A proactive malware identification system based on the computer hygiene principles," *Inf. Manage. Comput. Secur.*, vol. 15, no. 4, pp. 295–312, Aug. 2007.
[41] R. A. Wasniowski, "Multi-sensor agent-based intrusion detection system," in *Proc. 2nd Annu. Conf. Inf. Secur. Curriculum Develop. (InfoSecCD)*, 2005, pp. 100–103.
[42] C. J. Fung, J. Zhang, I. Aib, and R. Boutaba, "Robust and scalable trust management for collaborative intrusion detection," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, Jun. 2009, pp. 33–40.

MANSOOR NASIR received the master's degree in computer science from the Institute of Management Sciences, Peshawar, Pakistan. He is currently pursuing the Ph.D. degree with Islamia College University Peshawar, Peshawar. His research interests include information security, distributed systems and trusted computing, block chaining, edge AI, computer vision, and resource constraint hardware.

KHAN MUHAMMAD (Member, IEEE) received the Ph.D. degree in digital contents from Sejong University, Seoul, South Korea. He is currently working as an Assistant Professor with the Department of Software and a Lead Researcher with the Intelligent Media Laboratory, Sejong University. He has filed ten patents and published over 100 articles in peer-reviewed international journals and conferences in his research areas. His research interests include medical image analysis, such as brain MRI, diagnostic hysteroscopy, and wireless capsule endoscopy, information security, such as steganography, encryption, watermarking, and image hashing, video summarization, computer vision, fire/smoke scene analysis, and video surveillance. He is a member of the ACM. He is also an Editorial Board Member of the *Journal of Artificial Intelligence and Systems* and *Mathematics of Computation and Data Science*. He is also serving as a Professional Reviewer for over 70 well-reputed journals and conferences.

PAOLO BELLAVISTA (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science engineering from the University of Bologna, Italy. He is currently a Full Professor of distributed and mobile systems with the University of Bologna. His research activities span from pervasive wireless computing to location/context-aware services and from edge cloud computing to middleware for Industry 4.0 applications. He is also currently the Scientific Coordinator of a large H2020 big data innovation action called IoTwins about distributed digital twins for the manufacturing industry. He serves on the Editorial Boards of the IEEE Communications Surveys and Tutorials, *ACM Computing Surveys*, the IEEE Transactions on Network and Service Management, *Pervasive Mobile Computing* (Elsevier), *Journal on Network and Computing Applications* (Elsevier), and so on.

MI YOUNG LEE received the M.S. and Ph.D. degrees in image and information engineering from Pusan National University. She is a Research Professor with Sejong University. Her research interests include interactive contents, UI, UX, and developing digital contents.

MUHAMMAD SAJJAD received the master's degree from the Department of Computer Science, College of Signals, National University of Sciences and Technology, Rawalpindi, Pakistan, and the Ph.D. degree in digital contents from Sejong University, Seoul, South Korea. He is currently working as an Associate Professor with the Department of Computer Science, Islamia College University Peshawar, Peshawar, Pakistan. He is the Head of the Digital Image Processing Laboratory (DIP Lab), Islamia College University Peshawar, where students are working on research projects such as social data analysis, medical image analysis, multi-modal data mining and summarization, image/video prioritization and ranking, fog computing, the Internet of Things, virtual reality, and image/video retrieval under his supervision. His primary research interests include computer vision, image understanding, pattern recognition, and robot vision and multimedia applications, with current emphasis on raspberry-pi and deep learning-based bioinformatics, video scene understanding, activity analysis, fog computing, the Internet of Things, and real-time tracking.