IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 15, 2020, accepted August 28, 2020, date of publication September 3, 2020, date of current version September 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3021499

Alerts Correlation and Causal Analysis for APT Based Cyber Attack Detection

MEHRAN KHOSRAVI AND BEHROUZ TORK LADANI
Faculty of Computer Engineering, University of Isfahan, Isfahan 81746-73441, Iran
Corresponding author: Behrouz Tork Ladani (ladani@eng.ui.ac.ir)

ABSTRACT The advent of Advanced Persistent Threat (APT) as a new concept in cyber warfare has raised many concerns in recent years. APT based cyber-attacks are usually stealthy, stepwise, slow, long-term, planned, and based on a set of varied zero-day vulnerabilities. As a result, these attacks behave as diverse and dynamic as possible, and hence the generated alerts for these attacks are normally below the common detection thresholds of the conventional attacks. Therefore, the present approaches are not mostly able to effectively detect or analyze the behavior of this class of attacks. In this article, an approach for real-time detection of APT based cyber-attacks based on causal analysis and correlating the generated alerts by security and non-security sensors is introduced. The proposed method computes the infection score of hosts by modeling, discovery, and analysis of causal relationships among APT steps. For this purpose, a dynamic programming algorithm is introduced which works on alerts of each host separately and conducts a long-term analysis on the attack process to combat the outlasting feature of the APT attacks yet coping with a high volume of alert information. The proposed method is implemented and extensively evaluated using a semi real-world dataset and simulation. The experimental results show that the proposed approach can effectively rank hosts based on their infection likelihood with acceptable accuracy.

INDEX TERMS Advanced persistent threat (APT), attack process modeling, alerts correlation, causal analysis.

I. INTRODUCTION

The nature of computer attacks has been extensively changed by entering the cyber warfare to the field of classical wars. Modern computer attacks are usually targeted and act based on more definite and organized goals than conventional opportunistic attacks. Advanced Persistent Threat (APT) based attacks are considered as the most recent and significant attacks of this type which are usually designed and conducted by subversive organizations, governments, professional groups, and so on to reach their strategic goals. Modeling, analysis, identification, and confrontation with these attacks are new challenges in the field of computer security [21], [33].

There are two general approaches for detecting APT attacks: 1) approaches that seek for attack signatures (like attempting to infect the victim system, trying to communicate with C&C centers, and trying to destroy or steal information), and 2) approaches that seek to model the whole attack process. Modeling APT attacks by hidden Markov chain, decision making trees, and state machine based on attack steps and reciprocal behavior of attacker and defender are some examples of methods used in the second category. The first approach almost neglects the dynamic and intelligent nature of APT attacks, while the second approach better deals with the mentioned behaviors.

A major challenge in modeling the attack process of APT based attacks is that these attacks have a permanent diverse and dynamic behavior to outlast their presence in the infected hosts. Having diverse behavior means each attack uses a collection of new and various technics based on its objectives and execution conditions. This characteristic makes it hard to model APT based attacks only based on the known technics utilized by the previously identified attacks. Hence introducing approaches to cope with this problem is a challenge for detecting APT attacks. Having dynamic behavior on the other hand means the used technics can be changed based on the attack goals during the APT lifecycle. Since the period of persistence of an APT is often long, the attacker may change his/her attack technics by observing the reciprocal behavior of the victim system in attack time employing detection and defensive approaches used by the victim system.

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

Sometimes this dynamic behavior is managed and executed by communicating with the command and control (C&C) centers. The accurate modeling of this dynamic behavior can help better detecting APT based attacks.

As far as we know, considering the diverse and dynamic behavior of APT based attacks has not been already accounted for in this field accurately and fully by the previously proposed works. The main effect of diverse and dynamic behavior of APT attacks is to prevent their events from being fully recorded by existing security sensors or assigning low risk tags to the produced alerts. Therefore, the IKC associated with these attacks are sometimes not fully identifiable. An effective solution that we can offer to deal with diverse and dynamic behavior of APT attacks is to expand the range of input alerts. Hence we try to identify more IKC chains or complete the existing chains by considering alerts recorded by non-security sensors as well as security sensors and considering alerts with all risk levels (instead of just high risk alerts). Note that previous works mainly consider high risk alerts rooted in only security sensors. Our insight is that considering the accurate causal relationships among steps of attack in a long period and tracking them in time, while keeping the analysis in an appropriately high level of abstraction, considerably help detecting APT based attacks.

Therefore, in this article, we propose an attack process model based on correlating the produced alerts by various security and non-security sensors as a causal relationship sequence. For this purpose, alerts generated by different sensors are observed in a long time interval, and those that are more likely to form a sequence of APT attack steps are identified. The identification is hence based on the analysis of causal relationships of consecutive stages of the attack chain. Finally, the result is a score list that shows the probability of attacking different hosts in the system based on a normalized attack surface measure computed on the causal attack chains.

The proposed approach considers the challenges of detecting APT-based attacks including their diverse and dynamic behavior. These issues are less considered in other works mainly due to the lack of proper algorithms and complexity of extracting the relationships between attack steps. The proposed method can be implemented as an extension of a Security Information and Event Management (SIEM) system to analyze the events produced by the SIEM for scoring hosts via modeling and discovery of causal relationships among APT steps. The proposed method is implemented and extensively evaluated using a semi real-world dataset and simulation. The results of evaluations show that attack process modeling used in our approach can effectively rank hosts based on their infection likelihood with acceptable accuracy.

The rest of the paper is organized as follows: Section II introduces the basic concepts of APT based attacks. Section III reviews the related work. Section IV presents the architecture and steps of the proposed approach. In section V we conduct two different experiments to evaluate the proposed method and compare it with other related works to show its ability to rank the infected hosts and its precision. Finally Section VI concludes the paper.

II. APT BASED ATTACKS

The best definition for APT attacks can be obtained by the used title for this concept [1]:

- Advanced (A): APT attackers are sophisticatedly taught and well organized, their financial resources are supplied, and they use extensive intrusion technologies, data access, and data manipulation tools.
- Persistent (P): APT based attacks remain stable in the long term. Attackers seek a high-prioritized and specific goal instead of instant and temporary goals and keep their presence in the victim network for a long time (slow and permanent action).
- Threat (T): The attackers want to damage, disturb, and/or steal specific data or services.

Some samples of the discovered APT based attacks in recent years are Operation Aurora attack to Google infrastructure in 2009 and some other IT companies like Yahoo and Symantec; Stuxnet malware attack to Iran nuclear infrastructure and industrial systems of some other countries in 2010 which collected data and changed the victim system configuration (Siemens equipment); DarkHotel attack in 2014 (its new versions in 2018) that aimed at public services infrastructures to steal customer's information; and recently Kimsuky, APT27, and Lazarus or Vicious-Panda in 2020 who, according to OSINT, have used COVID-19-themed traps to target their victims [44].

The steps of an APT-based attack can be considered as follows:

- Reconnaissance: The attackers in this step collect data about the target organization's resources, employees, and their relationships with other institutions to obtain their mentioned goals.
- Delivery: This step includes making an email or a link of destructive code based on the collected data from the previous step, and sending it via a valid internet server or appraising the organization users to access to that link.
- Exploitation: After executing the destructive code by a victim, the attack path is made by the attacker and provides the possibility of exploitation. The next step is setting up a C&C center to manage all communication with the victim system.
- Operations: This step includes the presence of the attacker (e.g. via malware or peace of malicious code) in the victim organization network and exploitation for a long period.
- Data collection: The attacker collects the target data in this step through the obtained access permissions in previous steps. Also, other actions such as changing the settings or damaging systems are executed in this step.
- Exfiltration: In this step, the obtained information is packaged and coded precisely and sent to the

pre-determined servers. Moreover, the attacker tries to clean his/her presence trace on the victim system.

The above-mentioned steps of an APT-based attack are known as the Intrusion Kill Chain (IKC) [24]. IKC steps are normally common in APT attacks and what distinguishes APT attacks is the target of attack and the techniques used at each stage. Various, new, and sophisticated technics are used by the attackers to obtain the attack goals in each step of the IKC that will usually make a very light trace in forms of low-risk alerts. The attacker uses all types of deceptive, hiding, encrypting, and normal behavior imitation technics besides using the new and varied attacks to reduce his/her detection probability. This matter usually moves the generated alerts by security sensors below the detection threshold level (e.g. alerts with low risk).

III. RELATED WORK

Researches around APT based attacks can be classified into two categories: definition and detection of the APT attack, and defeating the APT attack. The researches on the first issue consider a precise definition of APT attacks, identifying the execution steps of these attacks, determining their characteristics [1], [7], [9]–[11], [21], [31] and finally detecting APT attacks [8], [14], [15], [27], [28], [29], [30], [33]. The works in the second category try to offer prevention measures or approaches for coping with these attacks by breaking the IKC or reducing the attack effects [13], [21]–[23], [33]. Note that approaches of this class of works are somehow dependent on the operational conditions and platforms (SCADA, Cloud, Cell Phone, Data Center, etc.).

Our work in this article is in the APT attack definition and detection category. Therefore, let us have a review of several related previous studies in this category. The selected works are those that not only are closely related to our work (from some aspects) but also are mostly pioneers and highly cited works.

Giura and Wang [1] after describing APT attacks and their related concepts have suggested a model for describing APT based attacks. They proposed a new approach to model APT attacks in the form of an attack pyramid. The aim of the attacker in the attack pyramid is at the peak of the pyramid, and the attack environments (physical environment, user environment, network environment, and other environments) make the pyramid sides. Joining the events in these environments makes an attack model. Events occurring in these environments are recorded by security tools such as firewalls and IDSs. Subsequently, these events are connected by applying correlation techniques. The appearance of an APT attack is detected by analyzing the integrated events based on the environmental conditions and the time context. The problem of the large data volume is raised in the proposed method while tackling the great amounts of events. The authors proposed to solve this problem using the MapReduce technic. A weakness of this work is that it does not consider the concept of kill chain in APT attacks, i.e. it just considers the recorded security events with high risk in the system, and the learning process is just on the environmental conditions and time contexts to integrate the events. Moreover, the presence of many irrelevant events in APT attacks which are detected in this model increases the false-positive error significantly.

Marchetti *et al.* [15] considered some characteristics of APT attacks such as being long-term, ordinary behavior imitation, and using encrypted communication as the challenges of APT attacks detection. The approach of this work is detecting and ranking hosts as victims of APT attacks. Therefore, an approach based on the detection of abnormal behaviors based on network traffic analysis by a special focus on the delivered packet's characteristics has been proposed. The focus of this work is on detecting the theft of information by APT attacks. Hence, the feature change settings and destruction of the victim's system information by these attacks are ignored in this work.

Niu and Zhang [11] considered some characteristics of APT attacks including their dynamic, unique, and time-consuming nature. The proposed approach in this work is based on modeling APT attack steps. Therefore, they introduced a model based on a complex network of the attack graph which encompasses the time concept too. Note that in prior works the attack graph did not cover the concept of time. As a weakness of this work, only the generated security events by the IDS are considered when switching between the defined modes of the complex networks. This is while the APT attacks' kill chain includes other events that are neglected in this approach.

Brogi *et al.* [7] considered some APT attack characteristics such as forming attack campaign, attack uniqueness, and long-term process of attack execution as the challenges of APT detection and modeling. The basic approach of this work is correlating the generated alerts by IDS in an attack campaign form. The focus of this work is only on IDS alerts and input/output traffic of a set of specified victims. This way, some specific generated alerts are correlated and other security and non-security alerts are neglected. This matter causes it to be unable to identify significant parts of the campaign that are made by probable attacks.

Wang *et al.* [14] considered some characteristics of APT attacks such as using the stealthy technics, zero-day vulnerabilities, the longtime taken to run an attack, and the issue of confrontation with the high volume of information as the challenges of detecting APT attacks. The insight of authors in this work is to identify C&C centers of APT attacks through differentiating between valid and non-valid DNSs and updating their lists. Thus, the characteristics of the infected DNSs are detected and are used to learn the algorithm. The main focus of this work is on detecting the attack event traffic characteristics and its main drawback is the lack of attention to both the fact that APT attacks use valid domains too, and C&C centers that rapidly change and sometimes are hierarchical.

Ghafir *et al.* [27] considered some APT attack characteristics such as being multistage, changing and scraping data, and

low precision of current detection approaches as the detection challenges of these attacks. APT attacks are considered as a four-stage process in this work including the stage of entering the victim system, C&C communication, reconnaissance, and exfiltration of the information. The considered APT attack in this work includes this four-stage scenario in a definite schedule; otherwise, if a part of this scenario (2 or 3 stages) is detected in the victim system (in a definite period), they are known as the sub-set or incomplete attacks. Based on this definition for APT attacks, a synthetic dataset is used to evaluate the suggested approach. The basis of the suggested approach in this work is using machine learning technics to detect the mentioned described four-stage chains or their subsets. A drawback of this work is that in building the model, only the network traffic is considered and no host-based event is used in the proposed model as part of the APT attack stages. Thus, the produced chain has a lost hook to consider the generated alerts inside a host which lose detecting many probable APT attacks.

Harikrishnan and Kumar [29] considered some characteristics of APT attacks such as their sophisticated structure, attack uniqueness, and long-term execution process as the challenges in modeling and detection of APT attacks. The approach of this work is based on using the generated logs by Splunk (a SIEM brand) and extracting characteristics of 100 APT attacks. These features are extracted by comparing the generated logs of the Splunk operation in two attack and normal modes. However, this work lacks consideration of the unique nature of the attack (in selecting the characteristics and symptoms). Furthermore, the events chain of APT-based attacks in this method limits the suggested approach's ability in detecting the probable attacks which leads to APT attack events in the long-term.

Milajerdi *et al.* [30] present HOLMES. This tool begins with host audit data (e.g., Linux audit or Windows ETW data) and produces a detection signal that maps out the stages of an ongoing APT campaign. HOLMES work with alarms that are generated for each host with audit logs. The idea in HOLMES is to use the information flow between low-level entities (files, processes, etc.) in the system as the basis for alert correlation. A drawback of this work is that for attacks that do not use or implicitly use the system calls, the proposed approach is ineffective and these attacks cannot be identified. Furthermore, attacks with multiple entry points cannot be identified by this approach.

Lajevardi and Amini [31] consider the fact that APT attacks are multi-stage, hybrid, long-term, and low-level. The proposed approach in their work is to use low-level interception and correlation of the operating system events with network events based on the semantic relationships that are defined between the entities in the system ontology. In this scheme, malicious events, especially the events that implicitly violate the security policies, are deduced and detected based on the event relations and the defined security policies. One of the main features of APT attacks is their permanency, but the approach presented in this article is not capable of analyzing long-term information flows and identifying attack scenarios due to the low-level relationships and the high volume of information. However, this approach can only be effective in identifying attacks that occur in the short term.

The approaches of the above-mentioned works and perhaps other related works are based on either detection of attack signatures or modeling the attack process. We believe that APT attacks have a dynamic behavior nature and exploit new and diverse vulnerabilities and attack techniques. Consequently, approaches based on identifying the attack signatures to counter these attacks are not efficient and/or general enough. The lack of proper algorithms and the complexity of extracting the relationships between attack steps are also major weaknesses of the attack process modeling-based approaches. Table 1 summarizes the reviewed works, their respective approaches, and their weaknesses.

TABLE 1. Summary of the works and the proposed approaches for different classes of APT attacks detection.

Works	Detection Category	Approaches	Weaknesses
Giura et al [1]	Modeling the attack process	Modeling APT attacks as attack pyramid	- Ignore the attack scenario - Only consider special resources events - Only consider high-risk events - Incomplete modeling
Marchetti et al. [15]	Detection of attack signatures	Detection of abnormal behaviors based on network traffic analysis	- Lack of generality - Ignoring the attacker's attempt to simulate normal behavior
Weina Niu et al. [11]	Modeling the attack process	Modeling APT attacks based on a complex network of the attack graph which encompasses the time concept too	- Only consider special resources events(IDS events) - Incomplete modeling
Brogi et al. [7]	Modeling the attack process	Correlating the generated alerts by IDS in an attack campaign form	- Only consider special resources events (IDS events) - Incomplete modeling
Xu Wang et al. [14]	Detection of attack signatures	Identification of C&C centers of APT attacks through differentiating between valid and non-valid DNSs	- Ignoring dynamic behavior of attacks - Ignoring the attacker's attempt to simulate normal behavior
Ghafir et al. [27]	Modeling the attack process	Create a 2-step model of attack based on network-only events	- Only consider special resources events (network events) - Incomplete modeling
Harikrishnan et al. [29]	Detection of attack signatures	Extracting and using characteristics of 100 APT attacks	- Ignoring dynamic behavior of attacks
Milajerdi et al. [30]	Detection of attack signatures	Using the information flow between low-level entities	- Lack of generality - Inefficiency of the proposed algorithms
Lajevardi et al. [31]	Modeling the attack process	Modeling semantic relationships that are defined between the entities in the system ontology	- Lack of generality - Only is effective in identifying attacks that occur in short term

The goal of this article is to present an approach for identifying APT attacks by attempting to complement and eliminate the weaknesses of attack process modeling approaches. As mentioned earlier, APT attacks occur in the IKC format, and what distinguishes APT attacks are the purpose and the techniques used at each stage of the attack. To this end, in the process modeling of the attack, causal relationships between the steps of the attack to create the IKC are considered.

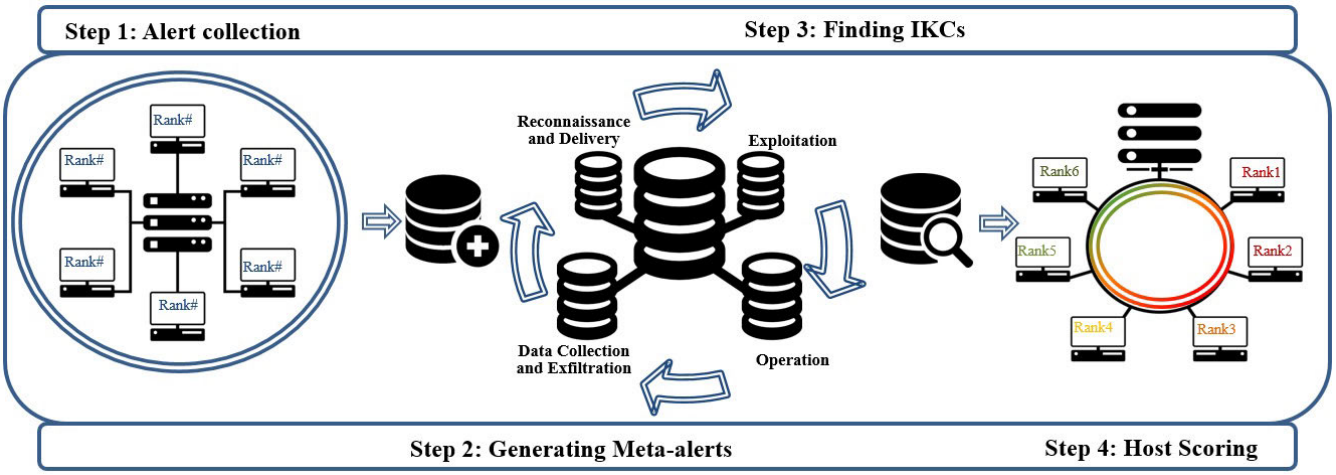

FIGURE 1. General sketch of the proposed approach for ranking the hosts by the likelihood of infection through APT based attacks.

IV. THE PROPOSED METHOD

In this section, we explain the proposed method for ranking hosts based on their likelihood to be exposed to APT attacks. For this purpose, we perform a causal analysis of the meta-alerts generated by both security sensors (like IDS and antivirus) and non-security sensors (like router and Windows log management system) to discover probable IKCs against the given hosts. Each IKC is assigned a score of being a part of an APT attack against the corresponding host and finally, a normalized attack surface value is computed for each host to show its rank of exposure to probable APT attacks. In the following, at first, the general structure and steps of the proposed method are formally explained and then we show how it can be formulated in form of a dynamic programming algorithm to perform the ranking process in real-time.

A. GENERAL STRUCTURE AND STEPS

APT based attacks use various attack techniques and exploit various vulnerabilities (diverse behavior), and their utilized techniques are subject to change during the time (dynamic behavior). However, what is nearly permanent in all APT attacks is the IKC that is constructed and managed by the attacker. By the way, the attacker in an APT attack tries to simulate the normal behavior of the system and prevents passing the thresholds of the detection systems. Therefore, there are a significant number of events about these attacks recorded by the non-security sensors that have been considered as ordinary events of the system with low risk. A model to describe these attacks will be effective when it can provide a proper and logical structure for representing the relationship between various events by connecting the recorded events of both security and non-security sensors. Our insight is that a suitable model can be designed to detect independent APT attacks when the basis of modeling is tracking the IKC. This model should correlate alerts with different risk levels based on causal relationships for each host to properly put the occurred alerts in the chain.

The general sketch of the proposed model is shown in Fig. 1. As shown in the figure, the process for ranking the score of infection of hosts by APT based attacks consists of four stages that are explained as follows.

Step 1: Alert collection

We aim to consider all of the system alerts (security and non-security) and correlating them based on their causal relationships. In most of the monitoring systems and proposed models, alerts are labeled with some risk severity levels e.g. high, medium, and low [45]–[48] and the focus of almost all works are only on high-risk alerts which are generated by the security sensors. However, it is to be remarked that in practice an IKC consists of a series of both security and non-security events that are almost with low-risk levels.

By the way, the approach of the previous works is mostly based on analyzing all generated alerts by collecting them in a single general model. Hence, the number of alerts for a given set of hosts is very high in the duration of time, and hence analyzing them is so problematic. To overcome this problem, alerts with low and medium risk levels are always ignored in previous works. Our idea is to consider and model events of each host separately instead of modeling the attack for the whole system. This way, the number of collected alerts for each separate host will be manageable. Hence, we will be able to consider all the generated alerts (security and non-security) with all risk levels.

Step 2: Generating Meta-alerts

The underlying principle of every SIEM system is to aggregate relevant data from multiple sources, identify deviations from the norm, and take appropriate action. We assume that there is a SIEM that generates alerts with different risk levels based on monitoring activities of security and non-security devices in the system. Our proposed method receives the

generated alerts from a SIEM and ranks hosts based on their likelihood of exposing with APT attacks.

For this purpose, the first step is to classify the received alerts into some meta-alerts corresponding to different steps of IKC using the given alert types. This is an abstraction we perform to reduce the high number of alerts generated by the SIEM while yet preserving the required features for doing our job. As an example, Table 2 shows how we can classify different generated alerts into four IKC step classes.

TABLE 2. Classification of alerts into IKC steps.

Alert class	Alert types	Meta-alert types
A^1	Refinement*, Reconnaissance*, Authentication*+, Port Sweep*+, Network Sweep*+, Password Guessing*, Identity Spoof*, Suspicious Domain Lookup*, Host Sweep*, Vulnerability Scan*, Self-Sign Certificate*, Brute Force*	Reconnaissance & Delivery
A^2	Command Injection*, File Inclusion*, Hijacking*, Update*+, Drive-by Download*, Cross Site Scripting*, Buffer Overflow*, Use After Free*, Bypass*, Application Exploit*+, Object Access*+, Memory Corruption*+, Shell Code*, SQL Injection*, ACL Policy Violation*, Privilege Change*+, ActiveX*, Script Injection*, Privilege Escalation*+	Exploitation
A^3	Command Injection*, Directory Traversal*+, Process Management*+, Data Access*+, Overloading Saturation*, Backup Management*+, Covert Channel*, Administration*+, Redirection*, Memory Corruption*+, Execution*+, Service Availability*+, Repair*+, Poisoning*, DoS*	Operation
A^4	Data Exchange*+, Botnet*, P2P Connection*+, Plain Sensitive Data*+, Non Standard Port*+, Download*+, Zone Transfer*, Information Leakage*, Replay Attack*, Protocol Anomaly*, Covert Channel*	Data collection & Exfiltration

* Security Sensors, +Non-security Sensors

According to Table 2, any of the four labels of Reconnaissance & Delivery, Exploitation, Operation, and Data collection & Exfiltration is selected to specify the steps where alerts are placed in the IKC. The alert types in Table 2 are in fact those that are produced by Ravin SIEM of Payam-Pardaz [34] using various security and non-security devices. To distinguish between alerts related to security sensors and non-security sensors in Table 2, the corresponding tags have been added to each alert type. Note that, the number of IKC steps may be considered more than four steps considering the nature of the system, the alert types produced by the underlying SIEM system, and the designated granularity for observing the IKC. Also, note that meta-alerts are generated regardless of the alert risk levels. As mentioned earlier, in contrast to some other similar works that consider only high-risk alerts for analysis, here we consider all risk levels because of the nature of APT attacks who try to be stealthy and work below the detection thresholds of the monitoring systems.

To formalize the model, let us first define the structure of a meta-alert in the system.

Definition 1: A meta-alert is defined as a six tuple (IP_{src}, $Port_{src}$, IP_{dst}, $Port_{dst}$, A^i, t) where IP_{src} and $Port_{src}$ are the source IP and source port of the alert respectively, IP_{dst} and $Port_{dst}$ are the destination IP and destination port of the alert respectively, A^i is the alert class, and t is the alert time where $T_0 \leq t \leq T$. T_0 is the time of the first alert in the alert observation window and T is the current time.

Now let us define the set of meta-alerts for a given host.

Definition 2: The set of meta-alerts for a given host h is defined as $MA_h = \{(IP_{src}, Port_{src}, IP_{dst}, Port_{dst}, A^i, t) \mid IP_{dst} = h\}$.

Step 3: Finding IKCs

The main goal of this step is to identify the causal relationships among the attack events in form of IKC. When one event is likely to be the antecedent for the other event in a possible IKC, we say they are in a causal relationship; the former event is the cause and the latter event is the effect. This may happen when all of the following conditions are met:

1) Co-location: Cause and effect alerts occur on the same host.
2) IKC ordering: Alert class index (IKC step) of the cause alert precedes the alert class index of the effect alert.
3) Temporal ordering: The cause alert precedes the effect alert in time.

This can be formally defined as follows.

Definition 3: Meta-alerts $a = (IP_{src}, Port_{src}, IP_{dst}, Port_{dst}, A^i, t)$ and $b = (IP_{src'}, Port_{src'}, IP_{dst'}, Port_{dst'}, A^{i'}, t')$ are correlated as a causal relationship from a to b ($a \rightsquigarrow b$) if all the following conditions are met:

Co-location: $IP_{dst} = IP_{dst'} \wedge Port_{dst} = Port_{dst'}$

IKC ordering: $i \leq i'$

Temporal ordering: $t \leq t'$.

For Meta-alerts within each host, the source address and destination address are the host address. For this Meta-alerts also considered "Null" for the field source port and destination port.

Now we can define the causal relation as follows.

Definition 4: A Causal Relation CR_h over the set of meta-alerts for a given host h (MA_h) is defined as $CR_h = \{(a, b) \in MA_h \times MA_h \mid a \rightsquigarrow b\}$

The causal relation helps us to define the concept of Causal Directed Acyclic Graph (CDAG) for a host h. Note that graphs in which vertices represent events occurring at a definite time, and edges are always connected earlier time vertices to later time vertices are necessarily directed and acyclic. This reflects our natural intuition of causality that means events can only affect the future and they never affect the past, hence we have no causal loops.

Definition 5: A Causal Directed Acyclic Graph for a host h is defined as $CDAG_h = (V,E)$ where $V \subset MA_h$ is the set of vertices, and $E = CR_h$ is the set of edges.

All directed acyclic graphs have a topological ordering, i.e. there is at least one way to order vertices such that direction of all edges is the same along with that ordering. Based on this concept, now we are ready to define an IKC for a given host as follows.

Definition 6: An IKC of host h with length n (IKC_h^n) is a Hamiltonian path in a $CDAG_h = (V, E)$ that is defined as follows:$(IKC_h^n) = v_1, v_2, \ldots, v_n$ where $v_i \in V$ *for* $i = 1, \ldots,$ n and there is a corresponding sequence of edges $a_1, a_2, \ldots, a_{n-1}$ such that $a_j = (v_j, v_{j+1}) \in E$ for $j = 1, \ldots, n-1$.

Note that a Hamiltonian path is a path in a graph that visits each vertex exactly once. According to definitions 6, APT attacks can take the form of a Hamiltonian path of at least 2 steps and up to the maximum step of the IKC. Each step can have multiple iterations in the path (not in a cycle or a loop) and meta-alerts of this path are casually correlated. For example, we can have an IKC chain with length 2 that contains so many meta-alerts. Note that althoogh an IKC is a Hamiltonian path, a Hamiltonian path is not necessarily an IKC.

It should be mentioned that checking whether a graph contains a Hamiltonian path is a hard problem, at the same time it is easy to perform such a check when the given graph is a DAG as in our case. To find Hamiltonian paths, we can use topological sorting. A topological sort or topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge from vertex u to vertex v, u comes before v in the ordering. It has been shown that any DAG has at least one topological ordering, and the usual algorithms for topological sorting have running time linear in the number of nodes plus the number of edges, asymptotically, i.e. $O(|V|+|E|)$ [35]. Therefore, if we assume that the number of meta-alerts of a given host is m, then $|V| = m$, $|E|$ is of order of m^2, and the complexity of finding all IKCs of the given host is of order $O(m + m^2) = O(m^2)$.

Step 4: Host Scoring

After finding all possible IKCs of different lengths for the given host, an assessment of the host infection probability should be made. Our insight is that hosts with more discovered IKCs, with more complete IKCs, and with IKCs that are created in a long time are more likely to be infected than other hosts. For this purpose, we define a measure to estimate the likelihood of host infection based on the mentioned intuition to rank the hosts in terms of probability of infection by APT attacks.

Definition 7: Let $IKC_h^n = v_1, v_2, \ldots, v_n$ and $v_j = (IP_{src}^{(j)}, Port_{src}^{(j)}, IP_{dst}^{(j)}, Port_{dst}^{(j)}, A^{i^{(j)}}, t^{(j)})$ *for all* $j = 1, \ldots, n$. We define Length Impact Coefficient (LIC) and Time Impact Coefficient (TIC) of IKC_h^n as follows:

$$LIC(IKC_h^n) = |\{A^{i^{(j)}} \mid for all j = 1, \ldots, n\}| TIC(IKC_h^n) = e^{\frac{t^{(n)} - t^{(1)}}{T - T_0}}$$

Note that the LIC of a given IKC shows the number of distinct values of Meta-alert types (IKC steps) in the IKC. LIC for a given IKC is higher as the number of distinct IKC steps is higher. This indicator helps considering APT attacks that are not fully recorded by the sensors (i.e. involve fewer IKC steps) in the infection assessment.

TIC also measures the time required to build the given IKC. Considering the nature of APT attacks which normally take place over a long time, the larger the TIC, the more likely the IKC to be an APT attack. However, any APT attack with any time interval (long or short) is considered by this indicator in assessing the level of infection.

Now we are ready to define a combined measure for evaluating the infection score of a given host.

Definition 8: Let H be the set of all hosts in the system, MA_h be the set of meta-alerts for a given host $h \in H$, and $\{IKC_h^*\}$ be the set of all discovered IKCs for host h with different lengths i.e. $\{IKC_h^*\} = \{IKC_h^{*(1)}, IKC_h^{*(2)}, \ldots\}$, then the Attack Surface of host h is defined as

$$AS(h) = \sum_{\pi \in \{IKC_h^*\}} LIC(\pi) \times TIC(\pi)$$

and the Normalized Attack Surface of host h is:

$$NAS(h) = \frac{AS(h)}{\sum_{h \in H} |MA_h|}$$

For ranking hosts in the system against APT attacks, it is enough to compute the normalized attack surfaces for all hosts. The hosts with higher normalized attack surfaces are more likely to be infected by some APT attacks.

B. REAL TIME ANALYSIS ALGORITHM

As mentioned earlier, the complexity of finding $\{IKC_h^*\}$ is $O(m^2)$ where m is the number of meta-alerts in the given host h. If we assume that the number of all hosts in the system is k, then the complexity of finding all IKCs is $O(km^2)$. This shows that the number of hosts has a linear effect on the complexity of the proposed method, and at the same time, it is not also reasonably so high. Therefore, we can assume that the main factor in the complexity of the proposed method is the number of meta-alerts produced by SIEM for each host in the system.

To have an estimation of the number of meta-alerts for each host, we can use the number of raw logs in the system. According to [39], an estimation of the number of all logs logged for 10K hosts over a period of one day in normal situations is about 1.5×10^9, and in peak time is about 0.5×10^9 which can be count totally as 2×10^9 logs. Therefore, it will be about 720 billion logs for each year for 10k hosts and hence about 72 million logs for each host. Note that all registered logs do not necessarily produce an alert. Alerts are usually generated by correlating the logs, policies of each organization, and the algorithms in the SIEMs in terms of three types of low, medium, and high risks. Regarding our practical observations with Ravin and other commercial SIEMS, the rate of alert per log is not more than 1% in most normal cases, although it may happen that due to considering special monitoring policies it reaches higher values. Assuming conservatively that 10% of the logs generate the alert, an average of 7.2 million alerts per host per year will be generated. This number means we have only less than 20K alerts per day, for each host.

Because of the nature of the APT attacks, the proposed method needs to access a long time log for the discovery of IKCs and computing the attack surface values. However, as we see, in the real world we have to deal with reasonably not so high newly generated alerts. Hence, although the complexity of the proposed algorithm is of the square root of the

total meta-alerts since the rate of generating alerts for each separate host is not so high, we can implement it as a real-time ranking system taking advantage of a dynamic programming approach.

Algorithm 1 Update-NAS

Input: ma_h

// Newly generated meta-alert for host h

Output: $NAS(h)$

//Normalized Attack Surface of host h

Maintain: A_h, B_h, AS_h, $TotalMA$

// Global data structures

$t = A_h$

// Finding new corrolations; t is a pointer to A_h

repeat

  Let $t = [ikc|IKClist]$

  Let $ikc = [ma|MAlist]$

  if $(ma_h \rightsquigarrow ma)$ **then**

    $ikc = [ma_h|ikc]$

    $AS_h = AS_h + LIC(ikc) \times TIC(ikc)$

    $TotalMA = TotalMA + 1$

    $NAS(h) = \frac{AS_h}{TotalMA}$

  end if

  $t = IKClist$

until $t \neq null$

// Checking it with no correlated yet ones

$found = no$

for $\forall ma \in B_h$ **do**

  if $(ma_h \rightsquigarrow ma)$ **then**

    $newikc = [ma_h, ma]$

    $A_h = [newikc|A_h]$

    $B_h = B_h \setminus \{ma\}$

    $found =$ yes

    $AS_h = AS_h + LIC(newikc) \times TIC(newikc)$

    $TotalMA = TotalMA + 1$

    $NAS(h) = \frac{AS_h}{TotalMA}$

  end if

end for

if $(found = no)$ **then**

  $B_h = B_h \cup \{ma_h\}$

end if

return $NAS(h)$;

The pseudo-code of updating the normalized attack surface for a given host h (i.e NAS_h) shown in Algorithm 1. This algorithm uses a dynamic programming approach and is done for the last generated alert of each host separately. The results are then integrated to rank the hosts in real-time. This way, the whole process also has a high potential of making it parallel and is very appropriate for real-time applications.

The algorithm also maintains some global data structures: list of all IKCs found so far for the given host h (A_h), the set of meta-alerts for the given host that are not yet correlated with any other meta-alerts (B_h),the current attack surface for the given host (AS_h), and the number of total meta-alerts used in all IKCs of all hosts so far.

For each generated meta-alert, first, the correlation of this alert is checked with the identified IKCs in list A_h. If a correlation is found, then the corresponding IKC will be completed. The correlation of the new meta-alert is then checked with the meta-alerts in set B_h to check new possible correlations with the met-alerts in this set. The meta-alert will be added to set B_h if no new correlation is identified. Each time that list A_h is updated, the value of NAS_h is recalculated and updated.

V. EXPERIMENTAL EVALUATION

In this section, we try to experimentally evaluate the proposed approach. We perform two experiments. The goal of the first experiment is to evaluate the applicability of the proposed approach in properly detecting hosts that are exposed to APT attacks in a semi-real environment. Furthermore, to evaluate the functionality of the proposed method in attack scenarios conducted in a long time, a second experiment is conducted via simulating the behavior of APT attacks in long periods. In experiments, we use several performance criteria to evaluate and compare the proposed approach to the other similar works.

One of the big challenges of evaluating our work and also other similar works is that providing real environments and even real datasets containing logs or events related to APT attacks is very hard. Hence designing a sound evaluation procedure is a real challenge. As far as we know, there is currently no datasets with labeled records of APT attacks. Specific APT related data is very limited and it is very hard to construct real-world datasets. This is mostly due to the nature of APT attacks: they are usually multi-stage and hybrid, designed for specific purposes, normally generate low-risk alerts and hence because of their low- risk level do not considered by detection and monitoring systems, and generally disappear or change after revealing or identifying their C&C centers. For these reasons most of the works on detecting APT based attacks use their own synthesized datasets instead of the real-world data [28]–[31].

To respond the mentioned challenge, we tried to provide a semi-real dataset and conducted simulation to generate the required meta-alerts. The semi- real dataset is a composition of a real-world APT attack free alert dataset with an extracted dataset from a real-world APT attack scenario description. We call it semi-real because it is synthesized from two real-world datasets. However, because the duration of the semi real alert scenario was not so long, we could not cover evaluating the proposed method while dealing with the long duration of APT attacks that is an important feature of these attacks. Therefore, in the second experiment, we try simulating the behavior of APT attacks for generating a series of simulated

events containing both normal events and APT attack events in a long time and with a large number of hosts.

A. FIRST EXPERIMENT: DETECTING HOSTS EXPOSED TO APT ATTACKS

In this experiment, we aim to evaluate the applicability of the proposed method in properly detecting hosts that are exposed to APT attacks. For this purpose, we first develop a semi-real dataset containing labeled APT attack IKC meta-alerts for some random hosts. After that, we perform a 10-fold cross-validation process. Through this process we find a proper attack surface threshold using the labeled meta-alerts in the training fold and then evaluate the method in discriminating between intact and exposed hosts using the achieved threshold in the test fold. Also, we compare the achieved performance results with some reported results of similar works.

1) GENERATING THE SEMI-REAL DATASET

For generating a dataset of meta-alerts containing labeled IKC and intact meta-alerts, we took some real-world alerts generated by a commercial SIEM to create an APT attack free meta-alert dataset and then randomly injected some APT attack meta-alerts derived from another dataset of real-world APT attack traces as labeled APT attack meta-alerts. The conducted steps for this process are explained in detail as follows:

Step 1 (Collecting APT attack free data):

We use a dataset of attack free meta-alerts produced by Ravin, a SIEM system developed by PayamPardaz Company [35]. Ravin is a commercial SIEM for collecting, analyzing, correlating, and reporting information about security and non-security equipment, servers, software, and other existing sensors in an organization. PayamPardaz released the Ravin product as a powerful SIEM for the deployment of Security Operation Centers (SOC) since 2012 and is currently the most popular domestic SIEM in the Iranian security market.

The considered dataset is the output of running Ravin in a company with 250 hosts containing 667K alerts generated for 3.1M investigated logs for a period of 3 days from September 23^{th} to September 26^{th}, 2018. Regarding the available information during and after the mentioned period and the type of functionality of the company, we sure that there was no APT attack in progress during this period. Therefore we assume that these alerts although may contain conventional attacks but is APT attack free. Hence we add a meta-alert tag to all of its alerts according to the procedure described in Step 2 in Section IV, as well as an ''intact'' label and use this dataset as our ground truth data.

The alerts generated by Ravin are in IDMEF format (the details of the format are described in RFC 4765). As an example, an alert produced by Ravin and its mapping to a meta-alert is shown in Fig. 2.

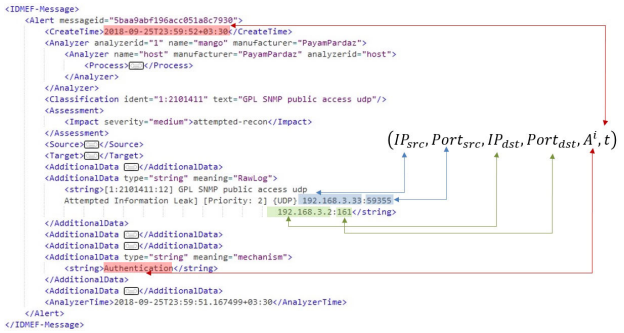

FIGURE 2. A sample alert produced by Ravin and the way of mapping it to a meta-alert.

Step 2 (Making APT attack meta-alerts):

The Transparent Computing (TC) program is a DARPA effort for developing technologies and an experimental prototype system to provide forensic and real-time detection of APTs as well as proactive enforcement of desirable policies [32]. We have used the report of the third TC adversarial engagement program (2018) named as TA5.1 Ground Truth Report Engagement 3 [32]. This program consisted of one scenario with multiple independent attackers where attackers consisted of two main groups, a nation-state, and a common threat and the goal of the nation-state attacker was to steal proprietary and personal information from the targeted company. The published report includes 27 APT attack scenarios conducted from April 6^{th} to April 13^{th}, 2018 including typical APT activities such as browser-induced drive-by initial compromises, backdoor injection, privilege escalation, internal reconnaissance, exfiltration of sensitive assets, and cleanup of attack footprints. In these attacks, sophisticated attack vectors such as reflective loading, web-shell capabilities, and in-memory module loading were used.

At first, we map the described scenario for all APT attacks in TA5.1 report into the data structures in our formalism. For this purpose, reading and analyzing each attack scenario manually, the meta-alerts for each arrack vector were extracted and represented as IKCs. Also for each extracted IKC, an IKC step-index was derived via analyzing the attack process and the defined classification process (see Table 2). Note that regarding the previous definitions and the existing information for each attack scenario in the report, mapping attack steps into meta-alerts and hence creating the corresponding IKC is not so hard. As an example, Fig.3 shows the schematic representation of the extracted IKC for the ''THEIA - Firefox Backdoor with Drakon APT In-Memory'' attack event. At each IKC step, a short description of the activities in that step as rationales for the performed mapping process has been shown.

Step 3 (Injecting TA5.1 APT IKCs into the Ravin meta-alert dataset):

We have a 7 days of attacks in TA5.1 scenarios and randomly injected the corresponding IKCs into the Ravin

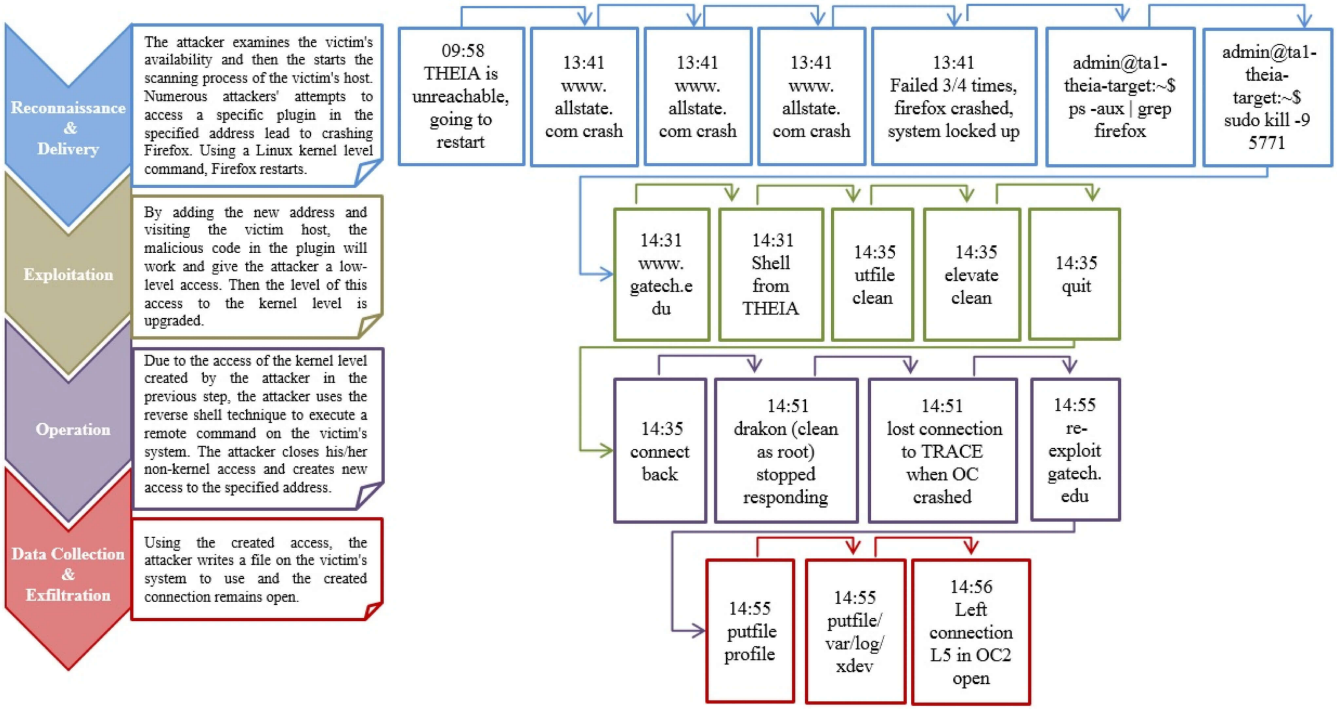

FIGURE 3. Schematic representation of the extracted IKC for the "THEIA - Firefox Backdoor with Drakon APT In-Memory" attack event.

meta-alerts dataset (which is 3 days). Assume that a host has been randomly selected and is a candidate to inject an attack vector scenario. The following steps are conducted for this purpose:

1) A random attack scenario in TA5.1 is selected and its corresponding IKC is derived.
2) The meta-alert time values in all the derived meta-alerts of the selected attack scenario are adopted. For this purpose, depending on the time frame in which the attack will take place on the selected host, we randomly generate the time for the TA5.1 attack vector and apply them to the records respectively. For example, if the attack is to take place within 2 days, we will generate a random time interval of 2 days for each meta-alert
3) The meta-alert destination IP address values in all the derived meta-alerts of the selected attack scenario are adopted. All of them are changed to the selected host IP address to which we are going to inject the attack.

2) EVALUATION PROCESS

After generating the semi-real dataset, we use it as our ground truth data to evaluate the proposed approach in properly ranking hosts that are more likely to expose to APT attack. We use a k-fold cross-validation approach to perform the evaluation process. The conducted evaluation demonstrates the performance of the proposed approach and the extent to which the results of the proposed method can be generalized and is independent of the training data. Note that in k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples in which one is used each time for testing and the other k-1 is used for training. This procedure is repeated, and all data is used exactly once for training and once for testing. Finally, the average result of this k validation is selected as the final estimate. Base on some researches [40]–[42] we select to use a 10-fold cross-validation method. For each iteration of the cross-validation process, we used 225 hosts (including 202 intact hosts and 23 exposed hosts) in the training phase and 25 hosts (including 22 intact hosts and 3 exposed hosts) in the test phase.

The NAS_h criterion for each host is used to rank the likelihood of exposing hosts to APT attacks. In the training phases, we try to find the optimal NAS_h for discriminating the exposed hosts from the intact hosts (i.e. the threshold value) based on the synthesized ground truth data. To do the optimization process we tried to maximize F1-score (The harmonic mean of precision and recall) by exploring the state space of the threshold value.

3) EVALUATION RESULTS

To show the effectiveness of the proposed approach, we conduct a sensitivity analysis experiment to analyze the effects of considering different security and non-security sensors and impact severity tags (risk levels).

The result of applying the optimum threshold value found in a sample round of the training phase and its corresponding test phase are shown in Fig. 4 in which all alerts generated by security and non-security sensors with any risk levels

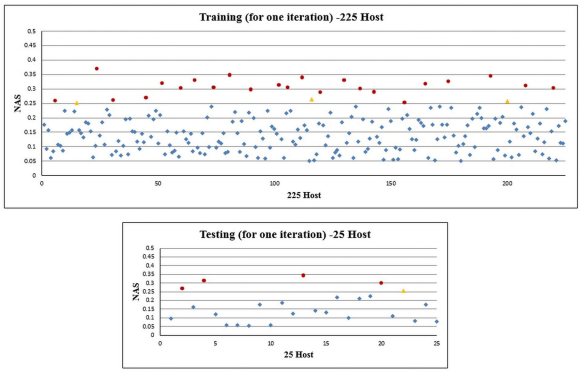

FIGURE 4. The classification results in one round of evaluation based on the values of NAS (considering alerts generated by both security and non-security sensors with all risk levels).

TABLE 3. The Results of evaluation with 10-fold cross-validation process.

Sensor type / Alert risk level	Number of alerts	Recall	Precision	Accuracy	FPR	F1-score
Security sensors / High risk alerts	170K	97.22%	44.30%	75.01%	30.56%	60.86%
Security sensors / All alerts	470K	95.37%	48.81%	79.07%	25.01%	64.57%
Security and non-security sensors / High risk alerts	520K	90.74%	57.98%	85.02%	16.44%	70.75%
Security and non-security sensors / All alerts	667K	87.10%	62.79%	86.39%	13.79%	72.97%

are considered. The circle (red) and rhombus (blue) samples are truly classified exposed and intact hosts respectively, and the triangle (yellow) samples are the misclassified samples according to the optimum threshold value found in this round of evaluation.

After achieving the optimum threshold values for each iteration, the average of all 10 iterations was obtained and was used to evaluate the performance in the test phase. The evaluation results including recall, precision, accuracy, FPR, and F1-score are shown in Table 3. Note that these criteria are the standard measures used in machine learning works [43]. The values in Table 3 are the averages of the values over 10 rounds of tests in the 10-fold cross-validation process. As we see, the highest precision in Table 3 is achieved when both security and non-security sensors and all alert types are used. Besides, the highest recall parameter is achieved when only security sensors and high risk alerts are used. This means considering non-security sensors and low risk alerts cost in lightly decreasing the recall (less sensitive detection) while much increasing the precision (more precise detection). However, yet the harmonic average of precision and recall (F1-score) is achieved when all sensors and alerts with all risk levels are considered. By the way, regarding high volume of alerts produced by SIEMS, normally it is preferable to have more precise systems than more sensitive systems.

B. SECOND EXPERIMENT: DEALING WITH LONG TIME ATTACK SCENARIOS

Our second experiment focuses on the long-term nature of APT attacks. As mentioned earlier, at the moment, we could not find any log or alert dataset to cover APT attack data in the long run (i.e. several months or years). Therefore, in this experiment, such a dataset is produced via simulation. For this purpose, we simulate the so-called "sleep & wake up" behavior of APT attacks that are typically used to extend the time of the attack [31].

The objective of conducting the second experiment is to investigate the ability of the proposed approach in 1) identifying hosts that are targeted by APT attacks over a long time, and 2) recognizing APT attacks before completion in as early stage as possible.

1) SIMULATING THE BEHAVIOR OF APT ATTACKS IN A LONG TIME

One of the important and distinguishing features of APT attacks is the long duration of these attacks that sometimes last from a few months to a year or even more. The reason for the long span of the attack is to prevent it from being detected by existing sensors and detection techniques. In the sleep & wake-up technique, the attacker performs part of the attack process, then enters the so-called sleep phase and does not perform any activity in the victim's set for a significant time, and then continues the attack process when it wakes up. This behavior of falling asleep and waking up intermittently over a long time causes the ineffectiveness of time-based window detection techniques. Also, this technique makes the detection methods that either have a computational overhead problem or are based on checking time-related close correlations ineffective.

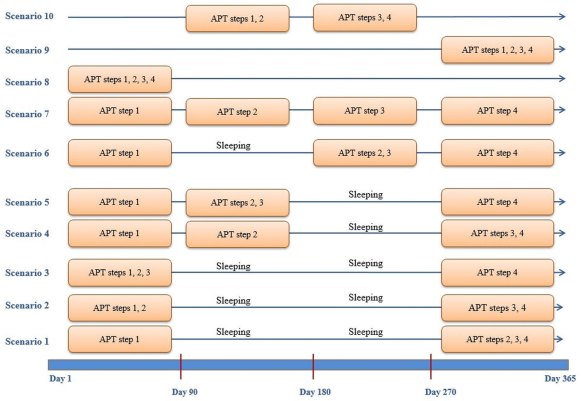

FIGURE 5. 10 scenarios for simulating APT attacks over a period of 365 days.

We use 10 different scenarios to model the technique of sleeping and waking up APT attacks according to Fig.5. In each of the 10 scenarios, the process of executing and sleeping the attacker is examined in a 365-day time frame. These scenarios can be traced every day for 365 days, but

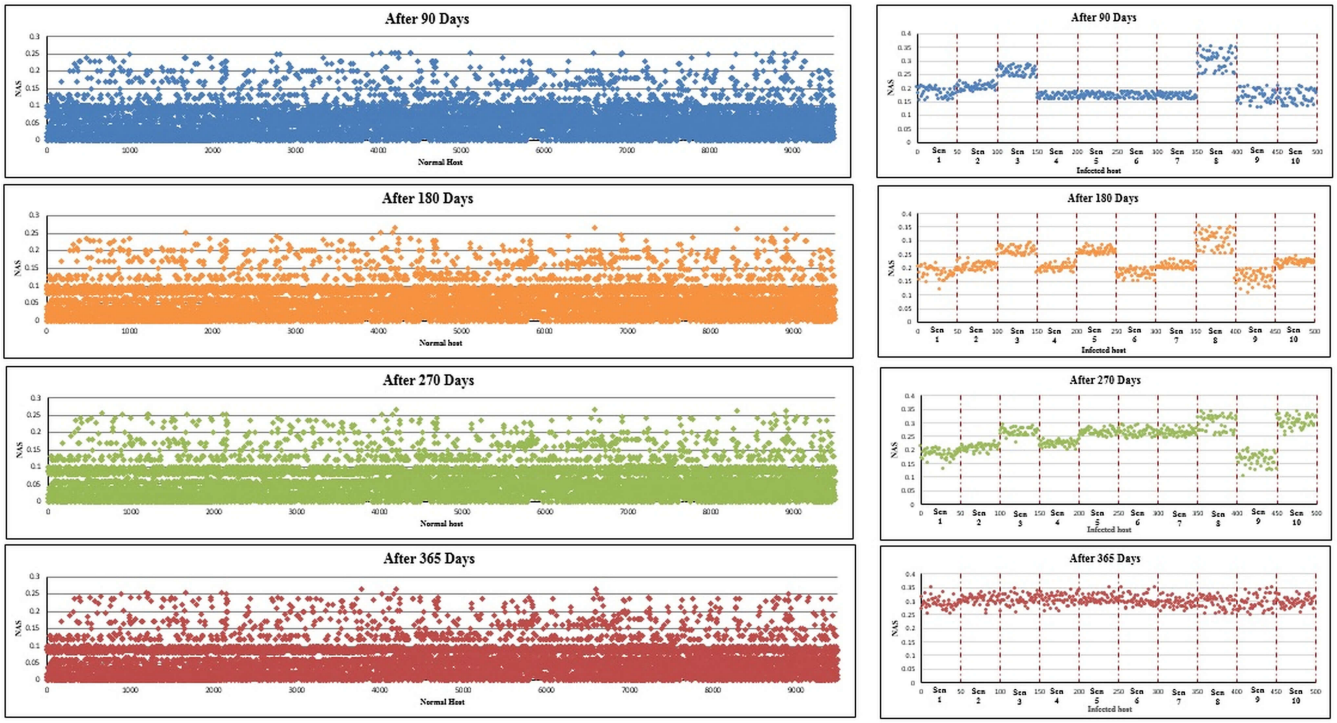

FIGURE 6. Comparison of NAS of intact and exposed hosts in a period of 365 days according to 10 different scenarios.

4 phases of 90-day intervals are used to model the injection process and also to compare the results.

To achieve the experiment objectives, we have written a Java code for simulating the behavior of attackers to generate meta-alerts for 10K hosts. The specifications of the system on which the simulation was performed are as follows:

- Windows 10 with Core i7 CPU - 3.40 GHz with 128GB of RAM

In the simulation process, we estimated that for 10K hosts, 60 billion logs will be recorded in a month, thus producing about 720 billion logs for a 12-month interval. Assuming 1% of this number will lead to producing meta-alerts, about 7.2 billion alarms were generated for 10K hosts. To produce the simulated alerts for each host we first generate a series of random meta-alerts for each host and then labeled APT attack related meta-alerts are injected into the series with a 365-day timeline according to the 10 mentioned scenarios over four 90-day time phases. For example, according to the fifth scenario, the attacker performs the first stage of the attack in the first 90 days. It then performs the second and third stages of the attack in the second 90 days. It then enters the sleep phase, and after a 90-day pause, performs the fourth stage of the attack in the last 90 days of the interval. At the end of each 90-day injection phase, the value of NAS is calculated and examined.

The process of injecting attacks is as described in the first experiment. We use the APT attack scenarios of the TA5.1 report collection to do the job gain. 9500 hosts out of 10K hosts are considered as intact and 500 hosts are intended for exposure to APT via injecting the attack vectors. There are 10 different scenarios for injecting attacks on these 500 hosts, so we applied a common scenario for every 50 hosts. Finally, the achieved NASs are used to compare the exposure and intact hosts under different scenarios at the end of each 90 days.

2) SIMULATION RESULTS

To compare the values of the NAS index for the hosts of the simulated set in different scenarios and situations over a period of 365 days we have shown the cumulative results of the experiments in Fig. 6. This figure consists of two parts: The graphs on the left side show the status of the NAS index for 9500 hosts in normal condition (in four 90-day phases). The graphs on the right side show the corresponding status of the NAS index for the 500 hosts with injected attack scenarios (in four 90-day phases). In each diagram in the left side, 10 different scenarios are separated by dashes.

As can be seen from Fig. 6, the NAS is not increased significantly as long as the attack is in its first and second stages, but as soon as the attacker enters the third stage of the attack and about 75% of the attack scenario is completed, it is increased significantly and its value is distinguishable from normal hosts. Hence the time window determines the correlation between alerts. As a result, no matter how much the attacker tries to prevent the attack from being detected using the sleep-wake technique, it will not affect the process of measuring the NAS index and attack detection.

The first objective of this experiment was to investigate the effect of conducting APT attacks in long runs on the number of detected host infections. Scenarios 1 to 9 can be considered for this purpose. As we see in all 9 scenarios, over time and with the completion of the attack, the NAS affected by the attack is increased and finally exceeds the threshold of 0.25 at the fourth 90-day phase.

The second objective of this experiment was to examine whether it is possible to identify APT attacks before completing all their steps. Scenarios 2 to 7, and 10 can be considered for this purpose. The results of the experiments show that after conducting the third step by the attacker, the NAS value approaches the threshold 0.25 and in most cases exceeds it which shows that the corresponding hosts are exposed to the attack.

C. COMPARISON AND DISCUSSION

Due to different conditions of the evaluation process and lack of standard datasets for APT attacks, a quantitative comparison of the detection performance of the proposed work with other similar works is not straightforward. However, assuming that all works are evaluated in an almost sound process, comparing the results of the proposed approach with other researches who have reported similar performance criteria makes sense. Fig. 7 shows the position of the proposed approach and three other similar works (who reported the required performance values) in the ROC (Receiver Operating Characteristic) space. Note that approaches who are placed near point (0, 1) in the ROC space have better performance than others. The position of the proposed approach in the ROC space (0.8710, 0.1379) shows that it brings in a better balance between recall and FPR parameters than other compared approaches.

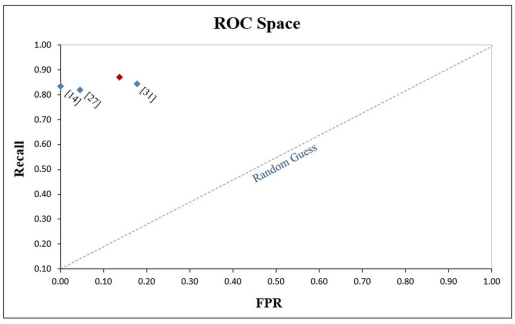

FIGURE 7. The position of the proposed approach and some similar works in the ROC space.

Table 4 numerically compares our proposed method and other similar works regarding the available reported performance indexes. Note that the mentioned results of other works are based on their reported values. However, although the experimental setup and environments of these works are different, assuming their achieved results are sound, we can use them to compare the works. As this table shows, because of the lack of required information, we are not able to precisely judge the superiority of a work over others although totally the achieved results of our proposed work are acceptable. However, it is necessary to mention that the works are different from some aspects (e.g. working real-time or offline, considering the whole alerts set simultaneously or per host, and so on) that makes it difficult to judge only based on the achieved quantitative performance indexes when comparing the works together. For this reason, we discuss some qualitative aspects of similar works and compare other works with ours as follows that show some important advantages of the proposed method over other similar ones:

TABLE 4. Comparison between the results of the proposed work and other similar works.

Approach	TPR (Recall)	FPR
MLAPT [27]	81.80%	04.50%
C&C-based [14]	83.30%	0
Context-based [1]	N/A	27.88%
Harikrishnan et al. [29]	99.00%	N/A
Lajevardi et al. [31]	84.21%	17.84%
The proposed work	87.10%	13.79%

- Balancing the early attack detection and accuracy: It is always desirable to detect APT attacks as earlier as possible before completely involving the victim. For this purpose, most approaches such as [7], [11], [14] try to detect APT attacks in the first phase of APT attack process (i.e. reconnaissance & delivery). This however almost leads to a high percentage of false alarms. As was shown in Fig. 6, the proposed approach tries to detect attacks continuously through all stages of the attack process. This approach that is more consistent with the nature of the APT attacks, brings in the highest available accuracy during the attack life cycle increasingly. Note that regarding the simulation results, the attack detection errors of the proposed approach at the first to the fourth stage of the attack process are 43.16%, 25.34%, 14.10%, and 10.47% correspondingly, that show a balance between early detection and high accuracy.
- Combating APT attack outlasting: APT attacks operate over a long and slow time frame. Approaches such as [11], [27], [29] that rely on time window to identify or emphasize the closeness of alerts and events in the correlation are almost unsuccessful in precisely identifying these attacks. At the same time, the long period of attack activities makes it necessary to deal with analyzing a large amount of information. According to [31], all existing approaches are ineffective in the face of outlasting features of APT attacks. The proposed approach uses a dynamic algorithm that slices the alerts based on separate hosts from one side and lets the desired time window be extended as necessary as possible from the other side. Therefore it can combat the outlasting feature of the APT attacks, yet successfully dealing with

a high volume of alert information using a divide and conquer approach.

- Ranking the hosts in terms of likelihood of being infected with APT attacks: In large organizations that work with large numbers of hosts, it is very important to identify which hosts are affected by APT attacks and how likely they are to be infected. Especially with limited manpower, organizations need to have priorities for considering hosts. The approach presented in this article taking advantage of NAS measure makes it possible to rank hosts in terms of their infection likelihood. Note that works who try to rank hosts are not either real-time or are not able to early detect APT attacks [11]. Moreover, as far as we know, there is no work to determine the likelihood of infection by APT attacks for each host. The work that ranks hosts like [15] only determines a priority list.
- Hybrid and wide risk range alert correlation: To precisely detect APT attacks we need to observe alerts generated due to the activities of various network and host sensors, i.e. we need to perform a hybrid alert correlation. By the way, we need to consider alerts with a wide risk range, rather than only high-risk alerts. The need for hybrid and wide risk range alert correlation has been overlooked in most previous works. As of this writing, there has been only one approach ([31]) that does hybrid alert correlation (i.e. considering both network and host alerts). Another problem with existing approaches is that they only consider high-risk alerts and ignore other medium- and low-risk alerts because they are reluctant to encounter large amounts of information in modeling [1], [7], [27]. As care about neither the alert origin nor the alert risk level in the proposed method, it can more accurately detect APT attacks.

VI. CONCLUSION

In this article, a method for modeling and detecting APT attacks is presented in which intrinsic characteristics of these attacks such as being stealthy, stepwise, slow, long-term, planned, and based on a set of varied zero-day vulnerabilities are considered. The proposed approach is based on causal analysis to correlate meta-alerts produced by SIEMs to discover probable IKCs against the given hosts. Each IKC is assigned a score of being a part of an APT attack against the corresponding host and finally, a normalized attack surface value is computed for each host to show its rank of exposure to probable APT attacks.

Two experiments were designed to evaluate the proposed approach. In the first experiment, by combining two real datasets from a commercial SIEM and the reported attack scenarios in the TA5.1 report, a labeled dataset from the actual APT attacks was generated to use it in evaluating the proposed approach. The results of the first experiment show the acceptable accuracy of the proposed model in real-time detection of APT attacks. In the second experiment, a fully simulated dataset of meta-alerts with injected APT attack vector meta-alerts was generated. Using this dataset we evaluated the functionality of the proposed method in different slow and long-term attack scenarios. The results of this experiment showed the ability of the proposed method in the detection of ongoing APT attacks especially when we face outsmarting behaviors in attack scenarios.

The proposed method can be implemented as an extension to SIEM tools to equip them with a real-time facility for the ongoing ranking of hosts in the organization according to their likelihood of infection to APT attacks.

This research has the potential for more work and can be continued in the future in different directions. For example, in addition to causal relationships in the correlation process, we can use information flow between alerts as well as contextual information to enrich the way of finding and scoring IKCs. This way, we can hope to do a more accurate detection with lower false positives. Furthermore, the proposed causal analysis method and the real-time algorithm can be adapted to different contexts such as fraud detection who deal with similar conditions. Note that in fraud detection problems we face similar underlying alert generation infrastructure as well as smart fraudsters who have similar APT attackers' characteristics.

ACKNOWLEDGMENT

The authors would like to thank their colleagues from PayamPardaz, especially Ahmad Reza Norouzi for assistance with collecting Ravin meta-alert dataset and also his comments that greatly improved the manuscript. They would also like to thank Amir Hosein Aliakbarian for helping them during the course of evaluation in this research.

REFERENCES

[1] P. Giura and W. Wang, "Using large scale distributed computing to unveil advanced persistent threats," *Sci. J.*, vol. 1, no. 3, pp. 93–105, 2012.
[2] B. Krekel, G. Bakos, and C. Barnett, "Capability of the People's Republic of China to conduct cyber warfare and computer network exploitation," US-China Econ. Secur. Rev. Commission, Washington, DC, USA, Res. Rep, 2009.
[3] B. Hartman, D. Martin, and D. R. Moreau, "Mobilizing intelligent security operations for advanced persistent threat," RSA Secur. LLC, Bedford, MA, USA, Tech. Rep. 11313-apt-brf, Feb. 2011, pp. 1–16.
[4] M. Antonakakis, C. Elisan, D. Dagon, G. Ollmann, and E. Wu, "The command structure of the Aurora botnet," Dambala, Atlanta, GA, USA, Tech. Rep. US-9686291-B2, Mar. 2010, pp. 1–32.
[5] B. Binde, R. McRee, and T. J. Oconner, "Assessing outbound traffic to uncover advanced persistent," SANS Tech. Inst., North Bethesda, MD, USA, Tech. Rep. JWP-Binde-McRee-OConnor, May 2011, pp. 1–35.
[6] Microsoft. (2011). *Microsoft Remote Desktop Protocol (RDP)*. [Online]. Available: http://bit.ly/UGGZCy
[7] G. Brogi and V. V. T. Tong, "TerminAPTor: Highlighting advanced persistent threats through information flow tracking," in *Proc. 8th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Nov. 2016, pp. 1–5.
[8] S. Siddiqui, M. S. Khan, K. Ferens, and W. Kinsner, "Detecting advanced persistent threats using fractal dimension based machine learning classification," in *Proc. ACM Int. Workshop Secur. Privacy Anal. (IWSPA)*, 2016, pp. 64–69.
[9] G. Brogi and E. Di Bernardino, "Hidden Markov models for advanced persistent threats," Ph.D. dissertation, Caen-Normandy Univ., Caen, Paris, 2017.
[10] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2nd Quart., 2019, doi: 10.1109/COMST.2019.2891891.

[11] W. Niu, X. Zhang, G. Yang, R. Chen, and D. Wang, "Modeling attack process of advanced persistent threat using network evolution," *IEICE Trans. Inf. Syst.*, vol. 100, no. 10, pp. 2275–2286, 2017.
[12] S. Chandran, H. P, and P. Poornachandran, "An efficient classification model for detecting advanced persistent threat," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Aug. 2015, pp. 2001–2009.
[13] A. Juels and T. F. Yen, "Sherlock Holmes and the case of the advanced persistent threat," in *Proc. 5th USENIX Workshop Large-Scale Exploits Emergent Threats*, 2012, p. 2.
[14] X. Wang, K. Zheng, X. Niu, B. Wu, and C. Wu, "Detection of command and control in advanced persistent threat based on independent access," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.
[15] M. Marchetti, F. Pierazzi, M. Colajanni, and A. Guido, "Analysis of high volumes of network traffic for advanced persistent threat detection," *Comput. Netw.*, vol. 109, pp. 127–141, Nov. 2016.
[16] I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Comput. Secur.*, vol. 48, pp. 35–57, Feb. 2015.
[17] Y. Wang, Y. Wang, J. Liu, and Z. Huang, "A network gene-based framework for detecting advanced persistent threats," *Proc. 9th Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput.*, Nov. 2014, pp. 97–102.
[18] A. Vance, "Flow based analysis of Advanced Persistent Threats detecting targeted attacks in cloud computing," in *Proc. 1st Int. Sci.-Practical Conf. Problems Infocommun. Sci. Technol.*, Kharkov, Ukraine, Oct. 2014, pp. 173–176.
[19] G. Vert, B. Gonen, and J. Brown, "A theoretical model for detection of advanced persistent threat in networks and systems using a finite angular state velocity machine (FAST-VM)," *Int. J. Comput. Sci. Appl.*, vol. 3, no. 2, pp. 41–64, May 2014.
[20] K.-F. Hong, C.-C. Chen, Y.-T. Chiu, and K.-S. Chou, "Ctracer: Uncover C&C in advanced persistent threats based on scalable framework for enterprise log data," in *Proc. IEEE Int. Congr. Big Data*, Jun./Jul. 2015, pp. 551–558.
[21] S. Singh, P. K. Sharma, S. Y. Moon, D. Moon, and J. H. Park, "A comprehensive study on APT attacks and countermeasures for future networks and communications: Challenges and solutions," *J. Supercomput.*, vol. 75, no. 8, pp. 4543–4574, Aug. 2019, doi: 10.1007/s11227-016-1850-4.
[22] R. Abreu, D. BobrowDave, and D. Burke, "Diagnosing advanced persistent threats: A position paper," in *Proc. 26th Int. Workshop Princ. Diagnosis*, 2015, pp. 193–200.
[23] A. R. Hern, A. C. VieIRA, and S. HHoumb, "Detection of advanced persistent threats using system and attack intelligence," *Proc. Emerg., 7th Int. Conf. Emerg. Netw. Syst. Intell.*, 2015, pp. 91–94.
[24] J. J. Mulligan, "Kill Chain, analysis of the 2013 target data breach," U.S. Congr., Senate, Committee Commerce, Sci., Transp., S. Rep.114-50, Mar. 2014.
[25] M. Parkour. (2013). *Contagio Malware Database*. [Online]. Available: http://contagiodump.blogspot.ca
[26] P. S. Ferrell, "Apt infection discovery using DNS data," Los Alamos Nat. Lab. (LANL), New Mexico, NM, USA, Tech. Rep. LA-UR-13-23109, 2013.
[27] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F. J. Aparicio-Navarro, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Gener. Comput. Syst.*, vol. 89, pp. 349–359, Dec. 2018.
[28] J. V. Chandra, N. Challa, and S. K. Pasupuleti, "A practical approach to E-mail spam filters to protect data from advanced persistent threat," in *Proc. Int. Conf. Circuit, Power Comput. Technol. (ICCPCT)*, Mar. 2016, pp. 1–5.
[29] V. N. Harikrishnan and G. T. Kumar, "Advanced persistent threat analysis using Splunk," *Int. J. Pure Appl. Math.*, vol. 118, no. 20, pp. 3761–3768, 2018.
[30] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "HOLMES: real-time APT detection through correlation of suspicious information flows," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 1137–1152.
[31] A. M. Lajevardi and M. Amini, "A semantic-based correlation approach for detecting hybrid and low-level APTs," *Future Gener. Comput. Syst.*, vol. 96, pp. 64–88, Jul. 2019, doi: 10.1016/j.future.2019.01.056.
[32] *Darpa-I2o/Transparent-Computing*. Accessed: Aug. 12, 2020. [Online]. Available: https://github.com/darpa-i2o/Transparent-Computing
[33] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2nd Quart., 2019.
[34] *Payampardaz Ravin SIEM Product*. Accessed: Aug. 12, 2020. [Online]. Available: https://payampardaz.com/en/ravin/
[35] O. Vernet and L. Markenzon, "Hamiltonian problems for reducible flowgraphs," in *Proc. 17th Int. Conf. Chilean Comput. Sci. Soc.*, Nov. 1997, pp. 264–267, doi: 10.1109/SCCC.1997.637099.
[36] S. W. R. Kevin, "Unique topological ordering," *Algorithms*, 4th ed. Reading, MA, USA: Addison-Wesley, 2011, pp. 598–599.
[37] S. A. Cook, "A taxonomy of problems with fast parallel algorithms," *Inf. Control*, vol. 64, nos. 1–3, pp. 2–22, Jan. 1985.
[38] E. Dekel, D. Nassimi, and S. Sahni, "Parallel matrix and graph algorithms," *SIAM J. Comput.*, vol. 10, no. 4, pp. 657–675, Nov. 1981.
[39] Aspiretss.com. 2020. *EPS Calculator–Aspiretss–Aspire Tech*. Accessed: Aug. 12, 2020. [Online]. Available: http://www.aspiretss.com/tools
[40] K. Ron, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. Int. Joint Conf. Artif. Intell.*, 1995, pp. 1–7.
[41] S. Borra and A. Di Ciaccio, "Measuring the prediction error. A comparison of cross-validation, bootstrap and covariance penalty methods," *Comput. Statist. Data Anal.*, vol. 54, no. 12, pp. 2976–2989, 2010.
[42] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statist. Surv.*, vol. 4, no. 0, pp. 40–79, 2010, doi: 10.1214/09-SS054.
[43] N. Japkowicz and M. Shah, *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge, U.K.: Cambridge Univ. Press, 2011.
[44] *APT Trends Report Q1 2020*. Accessed: Aug. 12, 2020. [Online]. Available: https://securelist.com/apt-trends-report-q1-2020/96826/
[45] Us-Cert.Cisa.Gov. (2020). *CISA Cyber Incident Scoring System | CISA*. Accessed: Aug. 12, 2020. [Online]. Available: https://us-cert.cisa.gov/CISA-Cyber-Incident-Scoring-System
[46] Security.ias.edu. (2020). *Priority And Severity Levels | IAS Security*. Accessed: Aug. 12, 2020. [Online]. Available: https://security.ias.edu/priority-and-severity-levels
[47] Docs.microsoft.com. (2020). *Insider Risk Management Alerts - Microsoft 365 Compliance*. Accessed: Aug. 12, 2020. [Online]. Available: https://docs.microsoft.com/en-us/microsoft-365/compliance/insider-risk-management-alerts?view=o365-worldwide
[48] *CIS Alert Level Information*. Accessed: Aug. 12, 2020. [Online]. Available: https://www.cisecurity.org/cybersecurity-threats/alert-level/
[49] P. W. Holland, "Statistics and causal inference," *J. Amer. Stat. Assoc.*, vol. 8, no. 81, pp. 945–960, 1986.

MEHRAN KHOSRAVI received the B.Sc. degree in software engineering from the Shahid Chamran University of Ahvaz, Iran, in 2010, and the M.Sc. degree from Azad University, Iran, in 2012. He is currently pursuing the Ph.D. degree in computer engineering with the University of Isfahan (UI), Iran. His research interests include software security and network security, malware analysis, and security risk management.

BEHROUZ TORK LADANI received the bachelor's degree in computer engineering from the University of Isfahan (UI), Isfahan, Iran, in 1996, the M.Sc. degree in software engineering from the Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 1998, and the Ph.D. degree in software engineering from Tarbiat Modares University, Tehran, in 2005. In 2005, he joined UI, where he is currently a Professor and the Dean of the Faculty of Computer Engineering. He is the author of more than 40 articles. His research interests include modeling, analysis, and verification of security in information systems, including software security (vulnerability detection and malware analysis) and soft security (computational trust and rumor control in social networks). He is a member of the Iranian Society of Cryptology (ISC) and the Editorial Board of the *International Journal of Information Security Science* (IJISS). He has been a Program Committee Member of the International ISC Conferences on Cryptology and Information Security (ISCISC). He was the Program Committee Chair of the ISCISC 2008 that was held in the University of Isfahan. He is the Managing Editor of the *Journal of Computing and Security* (JCS).