IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 23, 2019, accepted November 8, 2019, date of publication November 19, 2019, date of current version December 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2954092

Unknown Vulnerability Risk Assessment Based on Directed Graph Models: A Survey

WENHAO HE, HONGJIAO LI, AND JINGUO LI
Computer Science and Technology College, Shanghai University of Electric Power, Shanghai 200090, China

Corresponding author: Hongjiao Li (hongjiao.li@163.com)

This work was supported in part by the Opening Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security under Grant AGK2015005, in part by the National Natural Science Foundation of China under Grant 61403247 and Grant 61702321, and in part by the Project of Shanghai Science and Technology Committee under Grant 15110500700.

ABSTRACT Nowadays, vulnerability attacks occur frequently. Due to the information asymmetry between attackers and defenders, vulnerabilities can be divided into known and unknown. Existing researches mainly focus on the risk assessment of known vulnerabilities. However, unknown vulnerabilities are more threatening and harder to detect. Therefore, unknown vulnerability risk assessment deserves the widespread attention. To model the exploit process, directed graph models are applied to vulnerability risk assessment. And security metrics are used to quantify the exploitability of vulnerabilities. In this paper, according to the data source of nodes, related works of unknown vulnerability risk assessment based on directed graph models are divided into two types. One is based on network-level data, the other is based on system-level data. The former is to visualize the network status, while the latter is to reflect the running process of the system. The concept and purpose of these directed graph models are given at first. Then, these models are analyzed from three aspects, including advantages, flaws and solutions. After that, challenges and solutions of unknown vulnerability risk assessment based on directed graph models are given. Meantime, security metrics for unknown vulnerability risk assessment based on directed graph models are summarized and classified. Finally, future work directions of unknown vulnerability risk assessment are discussed from the perspective of techniques and application trends. Consequently, this paper can fill in the lack of current survey on unknown vulnerability risk assessment based on directed graph models.

INDEX TERMS Directed graph model, risk assessment, security metric, unknown vulnerability.

I. INTRODUCTION

With the continuous expansion of network scale, current network has the characteristics of large number of nodes, complicated structure, diversified protocols and data enrichment. Under this circumstance, network security is facing unprecedented challenges. To improve the integrity and stability of the network, risk assessment is proposed to evaluate the possible risks and provide a basis for network security. Data from HACKMAGEDDON [1] shows that vulnerability attack becomes one of the top 10 attack techniques. Directed Graph Model (DGM) is a major method for vulnerability risk assessment because it can visualize the network status and provide decisions for network hardening [2]. Meantime, DGM can reflect the running process of the system.

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

Vulnerability risk assessment based on directed graph models needs to accomplish both qualitative and quantitative tasks. For Known Vulnerability Risk Assessment (KVRA), vulnerability information can be obtained by vulnerability scanners, such as Nessus, Nmap, etc. Directed graphs can be generated automatically by existing tools. Meantime, standards such as Common Vulnerability Scoring System (CVSS) [3] can be directly used to quantify the exploitability of each known vulnerability. However, KVRA does not consider the situation that defenders may have less or no prior knowledge on vulnerabilities.

To solve this problem, the technology of Unknown Vulnerability Risk Assessment (UVRA) is proposed and should be given more attention because unknown (zero-day) vulnerabilities are harder to detect. Moreover, the threat and loss caused by this kind of attack are far more serious than known vulnerabilities. For UVRA, current researches often

set a time point to divide known vulnerabilities into known and unknown vulnerabilities because the latter is difficult to obtain in reality [4], [5].

Due to the reason that KVRA only focuses on known vulnerabilities, the major task of UVRA is to propose new directed graph models or improve existing models to simulate zero-day exploits. Meantime, for unknown vulnerabilities, there is no existing standard to convert vulnerability scores. Therefore, another major task of UVRA is to propose new security metrics to quantify the exploitability of zero-day vulnerabilities. In this paper, security metrics are divided into standard and specific metrics based on their versatility. Vulnerability risk assessment based on directed graph models requires the quantification of nodes and paths. So, specific metrics are further divided into three aspects, including node metrics, path metrics and probabilistic metrics. And they will be discussed later.

In this paper, according to the data source of nodes, related works of UVRA based on directed graph models are divided into two types. One is based on network-level data, the other is based on system-level data. The concept and purpose of these directed graph models are given at first. Then, these models are analyzed from three aspects, including advantages, flaws and solutions. Meantime, corresponding examples are given to facilitate understanding, and security metrics for UVRA based on directed graph models are summarized and classified. Next, challenges and solutions of UVRA based on directed graph models are given. At last, future work directions of unknown vulnerability risk assessment are discussed from the perspective of techniques and application trends. Directed graph models for UVRA are often inspired or extended from the models for KVRA. Therefore, if a directed graph model has related works on KVRA, and this model or its extension can be applied to UVRA, the development process of this model on KVRA will also be introduced.

The rest of this paper is organized as follows. In Sect. II, a brief introduction of Unknown Vulnerability Risk Assessment (UVRA) based on Directed Graph Model (DGM) is given. In Sect. III, DGM for UVRA based on network-level data is given. In Sect. IV, DGM for UVRA based on system-level data is given. In Sect. V, challenges and solutions of UVRA based on DGM are discussed. In Sect. VI, future work directions of unknown vulnerability risk assessment are discussed from the perspective of techniques and application trends. Finally, the conclusion of this paper is given in Sect. VII.

II. PRELIMINARY

Network security refers to systems that protect Internet connections, such as hardware, software, and data from security attacks [6]. Risk assessment is used to identify potential hazards/threats, which often describes risks in a quantitative manner and appropriately represents uncertainty [7]. The ultimate goal of risk assessment is to improve the integrity and stability of the network.

Vulnerability risk assessment is one of the techniques for network security. As mentioned above, vulnerability risk assessment can be divided into KVRA and UVRA. KVRA only focuses on known vulnerabilities. On the basis of KVRA, URVA considers the information asymmetry between attackers and defenders, and it focuses on the risk caused by zero-day exploits. The definitions of zero-day (unknown) vulnerability and zero-day exploit are given as follows.

Zero-day Vulnerability [5]: The detail of a zero-day vulnerability is unknown except that its exploitation potentially yields any privilege on the destination host and depends on three assumptions:

- *Network connection exists between source and destination host;*
- *Existing privilege on the source host;*
- *Destination host opens a remote access service.*

Zero-day Exploit [5]: Given a network composed of a set of hosts H, for each $h \in H$ *and* $x \in (serv(h) \cup priv(h))$*, a zero-day vulnerability is denoted by* v_x*. A zero-day exploit is the triple* $< v_s, h, h' >$ *where* $< h, h' > \in conn$ *and* $s \in serv(h')$*, or* $< v_p, h, h >$ *where* $p \in priv(h)$*. serv(h) and priv(h) respectively denote the services and privileges of hosts.*

The commonalities of zero-day vulnerabilities must meet one of the following conditions [8], [9]:

- Excluded by vulnerability scanners, such as Nessus, Nmap, etc.;
- Attackers know it, however, vulnerability databases such as National Vulnerability Database (NVD) have no record about it.

In UVRA, directed graph models provide the qualitative method to reflect network status. More specifically, they can find out the possible associations between vulnerabilities in a managed network. Moreover, they also help provide decisions for security solutions and hardening [2]. The definition of directed graph is given as follows.

Directed Graph: Directed graph $G = (V, E)$*, where V denotes the set of vertexes, E indicates the set of edges, and each edge represents the correlation between vertexes.* $|V|$ *denotes the number of vertexes in G, where* $V = \{v_1, v_2, \ldots, v_n\}$*.* $|E|$ *indicates the number of edges in G, where* $E = \{(u, v) | u \in V,\ v \in V\}$*.*

Technologies for KVRA cannot meet the requirement of zero-day vulnerabilities, which is mainly caused by three reasons. First, for zero-day vulnerabilities, the information owned by attackers and defenders is asymmetric. That is, attackers often know the existence of a zero-day vulnerability in advance. However, defenders will only know it after the attack occurs. Second, directed graph models for KVRA may need appropriate changes to adapt UVRA. Third, existing databases such as NVD ignore the impact of unknown network attacks. Therefore, they cannot accurately evaluate the security improvement in network hardening.

Nevertheless, directed graph models used in UVRA are often the extension of models applied to KVRA. Meanwhile, it has been mentioned in Sect. I that unknown vulnerabilities

TABLE 1. Comparisons between network and system level data.

Main Aspects of Comparison	Network-level Data	System-level Data
Acquisition process	Easy	Difficult
Community support	Yes	No
Tamper protection	Poor	Good

may be divided from the set of known vulnerabilities by setting a time point. The benefit of this method is that the information of these zero-day vulnerabilities can be used to verify the correctness of risk assessment because they are actually from known vulnerabilities. That is, UVRA based on directed graph models is inspired from KVRA. Therefore, if a directed graph model, which is used in KVRA at first, can be applied to UVRA after appropriate extension, the development process of both known and unknown vulnerability risk assessment will be given to explain the changes in this model.

According to the data source of nodes, related works of UVRA based on directed graph models are divided into two types. One is based on network-level data, the other is based on system-level data. Comparisons between these two data sources are given in Table 1.

Network-level data such as vulnerability information can be easily obtained from existing vulnerability databases. However, system-level data such as system call comes from the kernel of hosts and does not depend on network connectivity. In other words, it is hard to build a unified database for system-level data. Therefore, the acquisition process of network-level data is easier than system-level data. For vulnerability databases such as NVD, there exists dedicated communities or teams to provide support. However, system-level data often comes from a managed network, which may be only supported by the system administrator. In terms of tamper protection, network-level data can be crawled and parsed, so it is likely to be tampered maliciously. However, system-level data comes from hosts, which is the underlying data. Therefore, it is difficult to be tampered.

III. DGM FOR UVRA BASED ON NETWORK-LEVEL DATA

This section introduces directed graph models for Unknown Vulnerability Risk Assessment (UVRA) whose nodes are constructed by network-level data. Directed graph models that rely on network-level data include attack graph, resource graph and Bayesian network. Extended models of these directed graphs will also be discussed.

The rest of this section is arranged as follows. First, the concepts of these models are given, including the definitions and purposes. Next, the general process of applying these directed graphs to UVRA is given, and related works of these directed graph models will be analyzed. Finally, these directed graph models will be discussed from three aspects, including advantages, flaws and solutions.

A. ATTACK GRAPH (AG)

AG is used to find all attack paths that can reach the attack target by simulating the process of network attacks. In addition

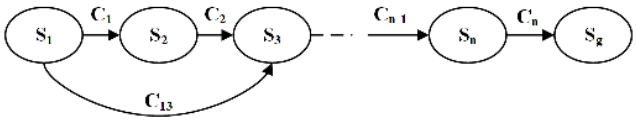


FIGURE 1. An example of original AG.

to the network connectivity provided by default, to construct an attack graph, the open services of each host in a managed network are required. Network connectivity between hosts belongs to the network layer of Open System Interconnection (OSI) model. And most open services of hosts such as *ftp* and *http* belong to the application layer of OSI model. Attackers often use vulnerability scanners to obtain vulnerabilities from these open services at first, and then use these vulnerabilities to achieve attack intention.

According to the specific meaning of nodes and edges, attack graphs are divided into state-based representation attack graphs and logical attack graphs. In state-based representation attack graphs (or called state-enumeration attack graphs), each node denotes a state, and each edge indicates the condition of state transition. In logical attack graphs, each node represents an exploit, and each edge denotes the correlation between exploits. Logical attack graphs are also called exploit dependency attack graphs. Risk flow attack graph and zero-day attack graph are the extended models of attack graphs. They both belong to logical attack graphs. These models will be analyzed in detail later. Here the definition of attack graph is given below at first.

Attack Graph [10]: An attack graph G is a directed bipartite graph $G = (E \cup C, R_r \cup R_i)$*, where E and C are the sets of exploits and security conditions, and the edges* $R_r \subseteq C \times E$ *and* $R_i \subseteq E \times C$ *respectively denote require and imply relations. More specifically, require relation denotes that the exploit cannot be successful unless corresponding condition is satisfied. And imply relation represents executing the exploit will satisfy corresponding condition.*

AG is first officially applied to vulnerability risk assessment in [11], and an example of original AG is given in Fig. 1. Each node denotes the state of a host, where S_g indicates the goal state to reach. And S_i shows the state before reaching the goal, where $i = 1, 2, \ldots, n$. Each edge represents the condition required for state transition. Although this work is the basis of vulnerability risk assessment based on attack graph, there still exists several problems, which are summarized as follows:

- Poor scalability and analysis complexity;
- Ignorance of the dependency between vulnerabilities, which may cause misleading results;
- The probability of measuring the success of a state transition depends on expert knowledge, in other words, this quantitative method is subjective.

To solve the poor scalability and analysis complexity of [11], in literature [12], the assumption of monotonicity is used to obtain a concise, scalable graph-based representation,

which indicates that the successful application of another exploit will never invalid the precondition of a given exploit.

In literature [13], [14], the problem of ignoring the dependency between vulnerabilities in [11] is solved by applying Bayesian network to attack graph. This model is called Bayesian Attack Graph (BAG). However, the general quantization method is not given in [13], [14], but it can be founded in [10]. The example of KVRA based on BAG will be introduced in Sect. III-C. The graph generation complexity of method proposed in [14] is $O(N^3)$, where N means the number of branches. Marginal Probabilities (MP) computing complexity is $O(2^n)$, where n indicates the number of variables. Genetic Algorithm (GA) complexity is $O(GN \log N)$, where G represents the number of generations, and N means the population size. In literature [10], Common Vulnerability Scoring System (CVSS) is introduced to quantify the exploitability of vulnerabilities, which tries to solve the subjectivity problem of quantification in [11].

Besides Bayesian network, in literature [15], another probabilistic directed graph model, called Hidden Markov Model (HMM), is used to construct probabilistic mapping between network observation and attack status. The definition of HMM is given after (2). The exploitability of each vulnerability (D_{v_i}) is calculated by (1):

$$D_{v_i} = AV \times AC \times Au \tag{1}$$

where AV denotes *Attack Vector*, AC indicates *Attack Complexity*, and Au represents *Authentication*. AV, AC and Au are the base metrics of CVSS. The probability distribution matrix of state transition is given in (2):

$$A = \{a_{ij}\} = \begin{cases} \dfrac{I_j}{\sum_{p=1}^{N} I_p}, & \text{if } S_i \overset{v_i}{\rightarrow} S_j \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where I_j denotes the weight that state S_i is converted to S_j via exploiting vulnerability v_i. The complexity of method in [15] is $O(N^2T)$, where N indicates the length of hidden states set, and T represents a constant.

Hidden Markov Model [15]: A HMM (λ) is a probabilistic model of time series, which is determined by probability matrix of state transition (A), probability matrix of observations (B), and probability vector of initial states (π). And it can be defined as $\lambda = (A, B, \pi)$, where

- $Q = \{q_1, q_2, \ldots, q_N\}$, $V = \{v_1, v_2, \ldots, v_M\}$, *where Q is a set of all possible states, V is a set of all possible observations, N indicates the number of possible states, and M represents the number of possible observations.*
- $I = \{i_1, i_2, \ldots, i_T\}$, $O = \{o_1, o_2, \ldots, o_T\}$, *where I is a state sequence, O is the corresponding observation sequence, and T represents the length of sequence.*
- *A is a probability matrix of state transition.*

$$A = [a_{ij}]_{N \times N}, \tag{3}$$
$$a_{ij} = P(i_{t+1} = q_j | i_t = q_i), \tag{4}$$

where $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, N$. a_{ij} is a probability of converting from state q_i at time t to state q_j at time $t + 1$.

- *B is a probability matrix of observations.*

$$B = [b_j(k)]_{N \times M}, \tag{5}$$
$$b_j(k) = P(o_t = v_k | i_t = q_j), \tag{6}$$

where $k = 1, 2, \ldots, M$, and $j = 1, 2, \ldots, N$. $b_j(k)$ is a probability of generating observation v_k under the condition of state q_j at time t.

- *π is a probability vector of initial state.*

$$\pi = (\pi_i), \tag{7}$$
$$\pi_i = P(i_1 = q_i), \tag{8}$$

where $i = 1, 2, \ldots, N$. π_i is a probability of being in state q_i at time $t = 1$.

In literature [16], the problem of ignorance on the dependency between vulnerabilities in [11] is also taken into consideration. Different from the idea of applying probabilistic directed graph models to attack graph, a model called Risk Flow Attack Graph (RFAG) is proposed to model network security by risk flow. The definition of RFAG is given below.

Risk Flow Attack Graph [16]: A RFAG can be defined as a tuple $RFAG = \{N, E, \tau, v, C, F\}$, where

- *$N = N_s \cup N_g \cup N_m$ stands for the set of nodes, where N_s indicates the initial capabilities of attackers, N_g represents the ultimate goal an attacker aims to achieve, and N_m denotes the multi-set of nodes η_i for which $\exists \varepsilon_1, \varepsilon_2 \in E | [(\eta_i \in pre(\varepsilon_1) \wedge (\eta_i \in post(\varepsilon_2))]$, pre() and post() respectively stand for the set of pre- and post-conditions. Each element in the node set has a value of **true** or **false**.*
- *$E \subset (N_s \times N_m) \rightarrow (N_m \times N_g)$ is the set of edges, which represents exploits in network.*
- *$\tau \subseteq N \times N$. An ordered pair $(N_{pre}, N_{post}) \in \tau$ if there exists an exploit edge $\varepsilon \in E$ that $N_{pre} \in pre(\varepsilon) \wedge N_{post} \in post(\varepsilon)$.*
- *$v : E \rightarrow Vuls$ is a mapping from an edge to its corresponding vulnerability. In a RFAG, an edge $\varepsilon \in E$ represents an exploit which is related to a certain vulnerability $Vul(\varepsilon)$. The metrics of the vulnerability will help determine the risk capacity and risk flow on edge.*
- *C is the risk capacity set of constants defined on E. The value of risk capacity $c(\varepsilon)$ is given by the CVSS base score of the vulnerability related to ε, that is, $c(\varepsilon) = Calc(Vul(\varepsilon) | T_{metric}(AV, AC, Au, C, I, A))$ and $c(\varepsilon) \in [0, 10]$. According to CVSS specifications, $c(\varepsilon)$ defines the maximum risk brought to the system once the correlated vulnerability is exploited.*
- *F is the risk flow set defined on E. Given an edge $\varepsilon \in E, f_e \in F$ denotes the amount of risk flow on edge ε, which indicates the actual risk when an exploit take place.*

In this work, the attack payoff and effort values of each path are calculated by (9) and (10):

$$payoff(p) = \sum_{\varepsilon \in p}[1 - (1 - CI)(1 - II)(1 - AI)] \quad (9)$$

where *CI* denotes *Confidentiality Impact*, *II* indicates *Integrity Impact*, and *AI* represents *Availability Impact*. They are also the base metrics of CVSS. p means an attack path.

$$effort(p) = \sum_{\varepsilon \in p}[20(AV * AC * Au)] \quad (10)$$

where *AV* denotes *Attack Vector*, *AC* indicates *Attack Complexity*, and *Au* represents *Authentication*.

If risk assessment depends on two or more factors, comprehensive evaluation method can be used to obtain more accurate result. First, a single-factor evaluation matrix should be calculated, which denotes the fuzzy relationship between risk factors and the final risk severity. Then, each factor will be assigned a weight to form a vector $A = \{a_1, a_2, \ldots, a_n\}$, where $0 < a_i < 1$, and $\sum_{i=1}^{n} a_i = 1$. At last, the final evaluation result B is computed by (11):

$$B = A \times R \quad (11)$$

where A is a weight vector, and R represents a single-factor evaluation matrix. The complexity of [16] is $O(N_{ch} \cdot N_s \cdot P)$, where N_s indicates source nodes, N_{ch} means the children nodes of N_s, and P represents paths.

The idea of attack graph with risk flow to model network security is also applied to industrial Internet of Things (IoT). In literature [17], attack graph is applied to industrial IoT to solve two problems, including the quantification and finding on attack paths. Graph generation complexity of this work is $O(n_{ep} \cdot n_p)$, where n_{ep} represents the number of elements in the set of attack instances, and n_p denotes the number of elements of the reachable precursor property. Attack reward *IMP* (or called attack payoff in [16]) is calculated by (12):

$$IMP(e) = 10.41 * [1 - (1 - CI)(1 - II)(1 - AI)] \quad (12)$$

where e represents the edge of an attack path. The equation to compute attack cost *EXP* (or called attack effort in [16]) is the same as (10). Equation (13), (14) and (15) are used to calculate the maximum loss flow:

$$lc(e) = IMP(e) - EXP(e) \quad (13)$$

where $lc(e)$ represents the difference between $IMP(e)$ and $EXP(e)$.

$$d_m(p) = \sum_{p \in P} lf^- \quad (14)$$

where d_m indicates the max flow loss in the preceding attack, p denotes an attack path, and lf^- represents the potential loss to next node.

$$\gamma = \frac{d_m(p)}{\sum_{e \in p} lc(e)} \quad (15)$$

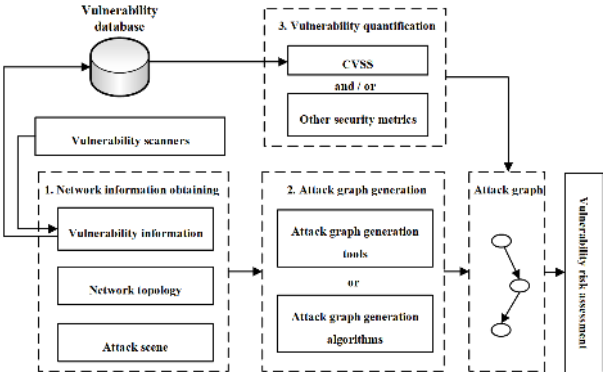


FIGURE 2. KVAR based on AG.

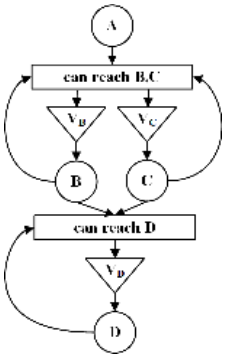


FIGURE 3. An example of MP graph.

where γ represents the ratio of d_m to the sum of loss capacity in the preceding attack. The impact of time measurement group and environment measurement group on quantization is not taken into consideration in this work.

The process of Known Vulnerability Risk Assessment (KVRA) based on Attack Graph (AG) can be summarized as Fig. 2. Common vulnerability scanners used for vulnerability risk assessment include Nessus, Nmap, etc. Detailed vulnerability information can be obtained from NVD.

In the early stage, attack graphs are generated by manual construction. This method leads to high error rate and poor scalability. In the previous studies [18], [19], Finite State Machine (FSM) is widely used in risk assessment, which is a kind of state-based representation attack graph. FSM can represent the transition between a limited number of states. However, it is not currently applied to unknown vulnerabilities, so it is not analyzed in detail here.

Nowadays, some tools support generating attack graphs automatically, such as MulVAL, TVA, Cauldron, NetSPA, etc. They are compared in Table 2. Multiple Prerequisite (MP) graph is a kind of attack graphs. MP graph contains three node types, and an example of MP graph is shown in Fig. 3.

First, each state node indicates the access level of attacker on a specific host. Second, each prerequisite node denotes an accessibility group or a credential. Third, each vulnerability instance node represents a specific vulnerability on a particular port. Each edge in MP graph indicates the

TABLE 2. Attack graph generation tools.

Name	Complexity	Vulnerability Database Used	Vulnerability Scanner Used	Attack Graph Type	Open Source
MulVAL	$O(n^2) \sim O(n^3)$	NVD	Nessus, OVAL	Logical attack graph	Yes
TVA, Cauldron	$O(n^2)$	NVD, OSVDB, CVE, Symantec DeepSight	Nessus, FoundScan	Logical attack graph	No
NetSPA, Firemon	$O(n \log n)$	NVD, CVE	Nessus, Nmap, NetViz	Multiple-prerequisite (MP) graph	No

TABLE 3. Development process of KVRA based on AG.

Literature	Year	Model	Cycles	Network Type	Future Development
[11]	1998	AG	Exist	Traditional network	Applying to enterprise-level (commercial or military) network risks
[12]	2002	AG	Eliminated by monotonicity	Traditional network	Validation of effectiveness on large networks
[13]	2008	BAG	Eliminated by monotonicity	Traditional network	Using dynamic Bayesian network with temporal metrics of CVSS
[14]	2012	BAG	Eliminated by monotonicity	Enterprise network	Improving the problem of graph generation, marginal probability computation and optimization
[16]	2015	RFAG	Eliminated by monotonicity	Traditional network	Security hardening and more network factors that affect security risk should be considered
[15]	2016	AG & HMM	Exist	Traditional network	Verifying the scalability of the method in a large-scale real network
[17]	2018	AG	Eliminated by monotonicity	Industrial IoT	Considering the impact of time factor and environment factor on quantization

relationship between nodes. These three node types are respectively represented as circles, rectangles and triangles in Fig. 3.

Comparisons of above works are shown in Table 3. From Table 3, it can be discovered that in KVRA based on attack graph, the elimination of cycles is often implemented by the assumption of monotonicity. In addition, the application of vulnerability risk assessment is gradually shifting from traditional network to enterprise network and industrial IoT. For example, the size of an enterprise network determines the complexity of assigning corresponding permissions and services to each host. In this condition, the vulnerabilities generated by the improper configuration of hosts may be exploited by attackers. Vulnerability risk assessment based on attack graph can visualize the network condition. Further, it provides the decision for optimal network hardening.

Researches mentioned above make a great contribution to network security, but they do not consider the attacks caused by unknown vulnerabilities. To solve this problem, an extended model of attack graph, called zero-day attack graph [4], [5], is proposed to compose with both known and zero-day exploits. Each path in this model is called zero-day attack path. A zero-day attack path is a multi-step attack path that includes one or more zero-day exploits [20]. The definition of zero-day attack graph is given below.

Zero-day Attack Graph [5]: Given the set of exploits of zero-day vulnerabilities E_0 and their pre- and post-conditions C_0, the set of exploits of known vulnerabilities E_1 and their pre- and post-conditions C_1, let $E = E_0 \cup E_1$, $C = C_0 \cup C_1$, and extend pre(.) and post(.) to $E \rightarrow C$ (as the union of relations). The directed graph $G =< E \cup C, \{< x, y >: (y \in E \wedge x \in pre(y)) \vee (x \in E \wedge y \in post(x))\} >$ is called a zero-day attack graph.

Next, the way to apply attack graph to UVRA will be presented. Then, related works on vulnerabilities risk assessment based on AG will be given.

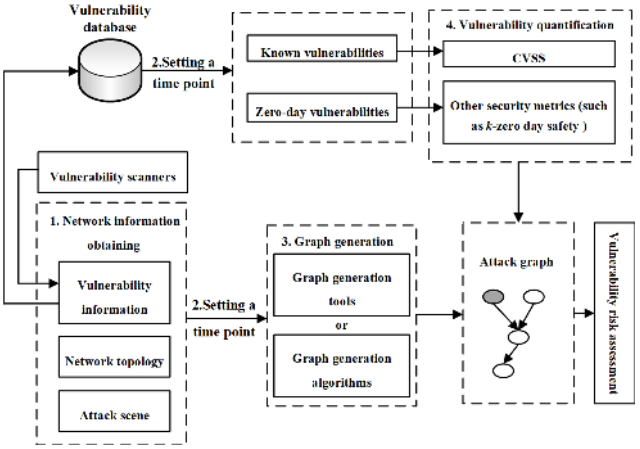


FIGURE 4. UVRA based on AG.

1) APPLYING AG TO UVRA

Unknown Vulnerability Risk Assessment (UVRA) based on attack graph is shown in Fig. 4. Compared with Fig. 2, it can be discovered that both KVRA and UVRA based on AG contain three steps, including information acquisition, graph generation, and vulnerability quantification.

The first step can be implemented with security tools, including vulnerability scanners like Nessus, and security sensors like Intrusion Detection System (IDS)/Intrusion Prevention System (IPS). The purpose of this step is to obtain necessary information as input to construct attack graph, including services and privileges of hosts, connectivity and vulnerability information.

Different from KVRA, the second step of UVRA based on AG is setting a time point to divide known vulnerabilities into known vulnerabilities and zero-day vulnerabilities because zero-day vulnerabilities are difficult to obtain in reality [4], [5]. For example, if the time point is set as 2018/12/31, vulnerabilities before 2018/12/31 are used as known vulnerabilities, and vulnerabilities after this time

TABLE 4. Previous works on the classification of security metrics.

Literature	Metric Family	Individual Metrics
[28]	Victimization	Existence, exploitability, impact
	Size	Vectors, machines
	Containment	Vectors, machines, vulnerability types
	Topology	Connectivity, cycles, depth
[32]	Path metrics	SP, NP, MPL, Normalized Mean of Path Lengths (NMPL), Standard Deviation of Path Lengths (SDPL), Mode of Path Lengths (MoPL), Median of Path Lengths (MePL)
	Non-path metrics	Weakest Adversary (WA), Network Compromise Percentage (NCP)
	Probabilistic metrics	State rank, cumulative score
	Bayesian network-based metrics	Logical *AND*, logical *OR*, the error rates of the IDS (false positive and false negative rates)
	Other metrics	*k*-zero day safety, network diversity
[31]	Host-based metrics (without probability values)	Attack cost, attack impact, Mean Time to Compromise (MTTC), structural important measure, Mean Time to Recovery (MTTR), the return on attack
	Host-based metrics (with probability values)	Probabilities of vulnerability exploited, probability of attack detection, probability of host compromised, CVSS
	Network-based metrics (path-based)	SP, NP, MPL, MMPL, SDPL, MoPL, MePL, attack resistance metric
	Network-based metrics (non path-based)	NCP, WA, vulnerable host percentage

point are regarded as zero-day vulnerabilities. Except for this method, in literature [21], suspicious activities whose signatures are not defined previously in Snort IDS/IPS are regarded as zero-day exploits. The former focuses on software or web applications, and the latter focuses on network packets.

The second difference between KVRA and UVRA based on AG is that for unknown vulnerabilities, there is no existing standard to convert vulnerability scores. Zero-day vulnerabilities are hard to measure because the process of discovering and exploiting vulnerabilities is less predictable [4]. To solve this problem, novel network security metrics are proposed, such as *k*-zero day safety [4], [5]. Further analysis of these related works will be introduced later.

In the domain of vulnerability risk assessment, security metrics are important because they are used to quantify the exploitability of vulnerabilities.

For known vulnerabilities, two kinds of scores need to be calculated, including individual score and cumulative score [22], [23]. Individual score represents intrinsic probability of an exploit being executed. It can be obtained by existing standards, such as CVSS, Common Configuration Scoring System (CCSS) [24] and Common Weakness Scoring System (CWSS) [25]. CVSS score is used in most current studies to measure the probability that a vulnerability is successfully exploited [10], [17], [26], [27].

Besides CVSS, in literature [28], a metric suite for attack graph is proposed, which is given in Table 4. In literature [29], to overcome shortcomings of path metrics, a complimentary suite of attack graph-based security metrics is proposed, including the Shortest Path (SP), the Number of Paths (NP) and the Mean of Path Lengths (MPL). In literature [30], model-based quantitative network security metrics based on attack graph are divided into five aspects. In literature [31], a systematic classification of existing security metrics based on network reachability information is proposed. Although these works make the contribution on the classification of security metrics, which are summarized in Table 4, the classification of security metrics for UVRA is still a blank. Therefore, the supplementary work is made in our paper, which can be seen in Table 7.

In literature [32], the calculation formulas of topology metrics (connectivity, cycle, and depth) are given below:

$$Connectivity\ metric = 10(1 - \frac{w-1}{d-1}) \tag{16}$$

where w is the number of subgraphs in the graph, and d indicates the total number of nodes in the graph.

$$Cycles\ metric = 10(1 - \frac{c-1}{d-1}) \tag{17}$$

where c represents the number of cycles in the graph.

$$Depth\ metric = \frac{10}{wd}\sum_{i}^{w} c_i(1 - \frac{s_i}{c_i - 1}) \tag{18}$$

where c_i means the number of nodes in different subgraphs, and s_i denotes the depth of each subgraph. In order to consider these three metrics comprehensively, the combined score is given as follows:

$$Combined\ score = 10\sqrt{\frac{\sum_{i=1}^{n}(s_i)^2}{\sum 10^2}} \tag{19}$$

where n is the number of considered metrics, and s_i indicates the individual score of each metric.

In literature [10], a probabilistic metric called cumulative score is first proposed. The relationship between exploits is first divided into two categories, including conjunctive and disjunctive [22]. After that, hybrid relationship is also taken into consideration. The definition of cumulative score is given below, which denotes the overall probability that an attacker can successfully reach and execute an exploit.

Cumulative Score [10]: Given an acyclic attack graph $G(E \cup C, R_r \cup R_i)$*, and any individual score assignment function* $p : E \cup C \rightarrow [0, 1]$*, the cumulative score function* $P : E \cup C \rightarrow [0, 1]$ *is defined as:*

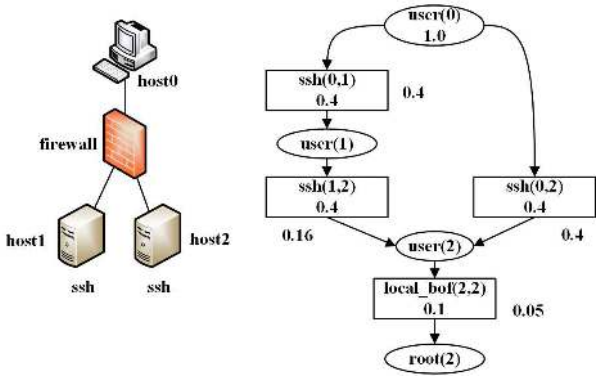


FIGURE 5. Network configuration and AG.

- $P(e) = p(e) \cdot \prod_{c \in R_r(e)} P(c)$
- $P(c) = p(c)$, *if* $R_i(c) = \phi$; *otherwise,* $P(c) = p(c) \cdot \oplus_{e \in R_i(c)} P(e)$ *where the operator* $\oplus$ *is recursively defined as* $\oplus P(e) = P(e)$ *for any* $e \in E$ *and* $\oplus(S_1 \cup S_2) = \oplus S_1 + \oplus S_2 - \oplus S_1 \cdot \oplus S_2$ *for any disjoint and non-empty sets* $S_1 \subseteq E$ *and* $S_2 \subseteq E$.

An example of network configuration and Attack Graph (AG) is shown in Fig. 5. Both *host*1 and *host*2 provide secure shell (*ssh*) service, and they are in the intranet. *host*0 represents an attacker who wants to penetrate into the intranet from the external network, so as to obtain the *root* privilege of *host*2. In this attack graph, ovals represent conditions, and rectangles denote exploits. Each condition represents a system state, and each exploit between hosts is reflected as a transition between system states. The decimals in rectangles, which are calculated by the base metrics of CVSS, represent the individual scores (or called exploitability) of vulnerabilities. Decimals next to rectangles denote the cumulative score of vulnerabilities. In Fig. 5, the exploitability of *ssh* and *local_bof* are respectively 0.4 and 0.1, where *bof* represents buffer overflow. The cumulative scores of node *ssh*(1, 2) and *user*(2) can be calculated as follows:

- $P(ssh(1,2)) = P(ssh(0,1) \times p(user(1)) = 0.4 \times 0.4 = 0.16$
- $P(user(2)) = P(ssh(1,2)) + P(ssh(0,2)) - P(ssh(1,2)) \times P(ssh(0,2)) = 0.16 + 0.4 - 0.16 \times 0.4 = 0.05$

However, for zero-day vulnerabilities, there is no existing standard to convert them into scores. They are hard to measure because the process of discovering and exploiting vulnerabilities is less predictable [4]. To solve this problem, a novel network security metric called k-zero day safety is proposed [4], [5]. The definition of k-zero day safety is given below.

k-Zero Day Safety [4]: Given the set of zero-day exploits E_0*, the definition can be that:*

- *a relation* $\equiv_v \subseteq E_0 \times E_0$ *such that* $e \equiv_v e'$ *indicates either* e *and* e' *involve the same zero-day vulnerability, or* $e =< v_s, h_1, h_2 >$ *and* $e' =< v_p, h_2, h_2 >$ *are true, and exploiting* s *yields* p. e *and* e' *are said distinct if* $e \not\equiv_v e'$;

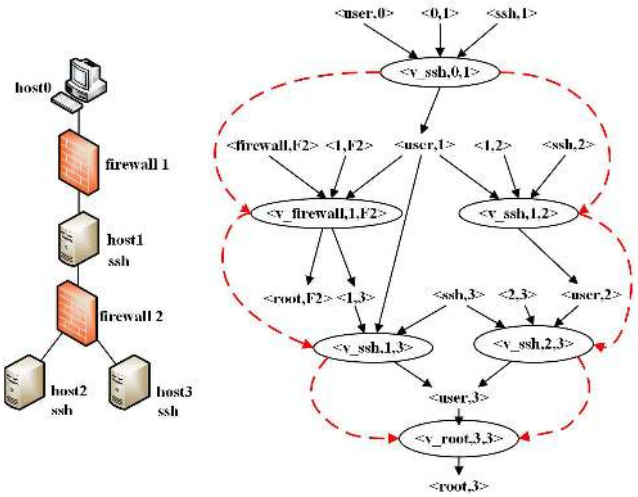


FIGURE 6. Network configuration and zero-day attack graph.

- *a function* $k0d(.) : 2^{E_0} \times 2^{E_0} \rightarrow [0, \infty]$ *as* $k0d(F, F') = max(\{|F''| : F'' \subseteq (F \triangle F'), (\forall_{e_1,e_2} \in F'')(e_1 \not\equiv_v e_2)\})$, *where* $|F''|$ *denotes the cardinality,* $max(.)$ *indicates the maximum value, and* $F \triangle F'$ *represents the symmetric difference* $(F \setminus F') \cup (F' \setminus F)$*; and*
- *for an asset* a, $k = k0d(a)$ *is used for* $min(\{k0d(q \cap E_0, \phi) : q \in seq(a)\})$, *where* $min(.)$ *denotes the minimum value. For any* $k' \in [0, k)$, a *is regarded as* k'*-zero day safe.*

This metric can be regarded as a node metric. The reason is that the core of this metric is to calculate the number of vulnerabilities (that is, nodes in the zero-day attack graph), which are required for compromising a network asset. For example, Fig. 6 shows network configuration and corresponding zero-day attack graph.

In Fig. 6, *host*2 and *host*3 are located in the intranet, and they only provide *ssh* service. *host*1 is located in the demilitarized zone (that is, the space between two firewalls). The *firewall*1 allows traffic to and from *host*1. The *firewall*2 allows traffic to and from *host*2, but only allows connections from *host*3. *host*0 is an attacker in the external network who tries to obtain the *root* privilege of *host*3. For known vulnerabilities, both vulnerability scanner and attack graph will lead to the same conclusion that the network configuration in Fig. 6 is secure [20]. However, for zero-day vulnerabilities, there are two attack paths for attackers to reach the target (that is, the *root* privilege of *host*3) in Fig. 6, which are described by red dotted lines. One is obtaining the *user* privilege of *host*1 at first, and then the attacker uses the *host*2 as a Jump Server to indirectly attack *host*3 by *ssh* service. The other is that the attacker uses the *ssh* service of *host*2 and *host*3 to directly attack *host*3 after achieving the *user* privilege of *host*1.

Here the network asset $A = \{< root, 3 >\}$. The left attack path contains three distinct zero-day vulnerabilities.

According to the definition, exploits of $< v_{ssh}, 0, 1 >$, $< v_{ssh}, 1, 2 >$ and $< v_{ssh}, 2, 3 >$ in the right attack path will be counted as one exploit because they involve the same zero-day vulnerability (*ssh*). So, it can be discovered that $k0d(A) = 2$, where $k0d()$ is the function to calculate k-zero day safety.

Although this metric makes a great contribution to evaluating zero-day attacks, it has following problems at the same time:

- This metric simply calculates the number of vulnerabilities required for destroying network assets. Meantime, the correlation between vulnerabilities and the impact of known vulnerabilities on UVRA are not taken into consideration;
- It is difficult to determine exact value of k;
- It assumes the existence of a complete attack graph, however, this assumption is difficult to establish in large networks [20].

In order to solve these problems, existing researches mainly focus on two aspects:

- Improving the calculation ability of k-zero day safety;
- Improving the metric system of k-zero day safety.

In literature [20], the exact value of k is obtained by calculating the lower bound and upper bound of k. Corresponding processes to calculate these two bounds are respectively summarized as Fig. 7 and Fig. 8.

In Fig. 7, C_i denotes the set of initial conditions, E^* indicates the set of known and zero-day exploits, l is an integer representing the desired lower bound of k, c_g means the goal condition, C denotes the set of conditions, E indicates the set of exploits, C_{new} represents the set of newly satisfied conditions, $\pi(c)$ means the mapping which associates each condition with a set of attack paths leading to it, $e \in E$, c represents the pre- or post-condition of an exploit, and G denotes the partial zero-day attack graph.

In Fig. 8, R_r and R_i respectively denote *require* and *imply* relations. More specifically, *require* relation denotes that the exploit cannot be successful unless corresponding condition is satisfied. And *imply* relation represents executing the exploit will satisfy corresponding condition. $zdu(c)$ indicates the number of required exploits for reaching initial conditions, $zdu(e)$ means the number of distinct zero-day vulnerabilities in an edge, and u represents the upper bound of k.

The efficiency of graph generation is improved by using on-demand method and reusing the partial attack graphs that have been generated in the decision process. This method can apply to large networks because it spends less than 20s to build an attack graph of nearly 90,000 nodes. Processing time and percentage of nodes are used as performance metrics of algorithms. The shortcoming is that they all focus on improving the computational efficiency of k-zero day safety, but no one considers the correlation between vulnerabilities. The future development of this model is to improve the approximate algorithm for ranking the partial solutions.

Compared with [20], in literature [4], a new heuristic algorithm, which computes k-zero day safety as the shortest path in a directed acyclic graph, is proposed for efficiently computing the metrics in special cases. This work relies on the assumption that both conjunctive relationship between conditions and the similarity between zero-day exploits are mostly limited to each host or each small group of hosts. The problem of determining the value of k is converted to finding the shortest path in the set of zero-day exploits.

To improve the security metric system of zero-day vulnerability, in literature [33], network diversity is modeled as a security metric so as to evaluate the robustness of networks against potential zero-day attacks.

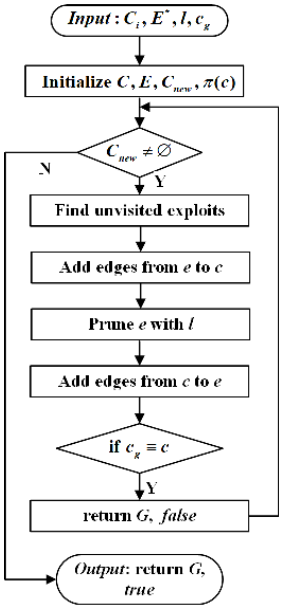


FIGURE 7. Obtaining the lower bound of k.

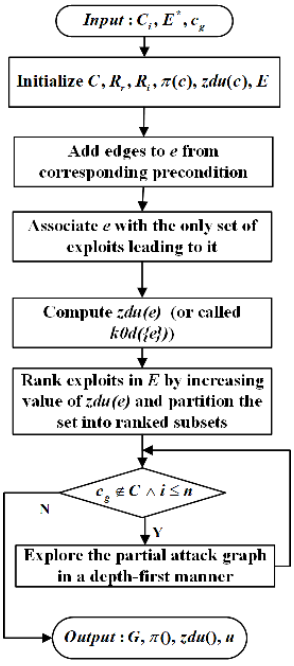


FIGURE 8. Obtaining the upper bound of k.

In literature [34], *tolerance* is defined as a metric to capture the required zero-day attack effort. However, they only consider individual zero-day weakness under different targets and ignore the multiple zero-day exploits.

Inspired from k-zero day safety, in literature [9], three security metrics are proposed to improve the metric system of k-zero day safety, including k-zero day safety of a zero-day vulnerability, length of the k-zero day safety, and exploitability of k-zero day safety. Length of the k-zero day safety can be divided into node metrics because it indicates the zero-day attack path which includes the minimum number of zero-day vulnerabilities. k-zero day safety can be divided into probabilistic metrics because it represents the exploitability level of each attack path. In this work, the impact of known vulnerabilities on evaluating the risk of zero-day attacks is taken into consideration, and the risk of zero-day attacks can be differentiated. The formulas used in risk assessment are as follows:

$$Exploit(Path) = \frac{1}{PL} \times \sum_{i=1}^{PL} Exp(V_i) \tag{20}$$

where $Exp(V_i)$ denotes the exploitability of vulnerability V_i, and PL represents the length of the k-zero day safety path. Equation (20) means the exploitability of each attack path which leads to the attack goal.

$$Prob(vul) = \frac{1}{PL} \times \frac{1}{KZS} \times \frac{Exploit(KPath)}{10} \tag{21}$$

where KZS indicates the k-zero day safety, and $KPath$ denotes k-zero day safety path. Equation (21) represents the probability of exploiting each vulnerability.

$$Risk(V_i) = Probability(V_i) \times Impact(V_i). \tag{22}$$

In (22), $Impact(V_i) = 6.4$. Exploitability of each zero-day vulnerability is set as 10. Therefore, the shortcoming of their work is that the exploitability and impact of each zero-day vulnerability are set to a constant value, which cannot reflect the actual situation well.

2) DISCUSSION ON AG FROM ADVANTAGES, FLAWS AND SOLUTIONS

Attack graph is a powerful tool to access network security and provide decision for network hardening. It can be used to identify undesirable activities caused by attackers. Meantime, it can also help security administrators understand whether given critical resources can be compromised through multi-step attacks. The advantage of attack graph is that it can be automatically generated by tools.

However, as a qualitative model, attack graph also has some flaws. First, attack graph only reflects whether a managed network is secure or not, it cannot quantify the threat level of network. Second, attack graph may include cycles, which is inconvenient to use security metrics to quantify later. Third, attack graph cannot deal with the uncertainty of network attacks.

Zero-day attack graph fixes the flaw that only considering known exploits. In this model, both known and zero-day exploits are considered. However, in terms of complexity, zero-day attack graph is comparable to traditional attack graph. The reason is that the number of added zero-day exploits (which depends on the number of remote services and privileges) on each host should be comparable to the number of known vulnerabilities [4]. That is, vulnerabilities depend on the size of the network. As the network density becomes larger, there is a greater likelihood of vulnerabilities in the network [35].

The assumption of monotonicity is often used to eliminate the cycles in attack graphs. Besides this method, in literature [23], cycles are divided into three types. The first type is that cycles can be removed directly. The second type is that cycles can be broken. The third type is that cycles can neither be removed or broken. This method is complex in practice.

To address the uncertainty of network attacks, Bayesian network is often applied to constructing attack graph, and the corresponding content will be given in Sect. III-C.

B. RESOURCE GRAPH (RG)

Resource Graph (RG) is used to reflect the strategy that may be chosen by attackers to reach the final condition (that is, a critical network asset) with the least effort. Each node in resource graph indicates a zero-day exploit [33]. Each edge represents the dependency between zero-day exploits. Resource graph focuses on remote access services, such as *http*, *rsh* and *ssh*. These services all come from the application layer of OSI model. The definition of resource graph is given below.

Resource Graph [33]: Given a network composed of a set of hosts H, a set of resources R with the resource mapping $res(.) : H \rightarrow 2^R$, a set of zero-day exploits $E = \{< r, h_s, h_d > | h_s \in H, h_d \in H, r \in res(h_d)\}$ and the collection of their pre- and post-conditions C, a resource graph is a directed graph $G(E \cup C, R_r \cup R_i)$, where $R_r \subseteq C \times E$ and $R_i \subseteq E \times C$ are the pre- and post-condition relations respectively.

1) APPLYING RG TO UVRA

Resource graph is similar to attack graph in information acquisition and graph generation. The reason is that resource graph is syntactically equivalent to attack graph, but resource graph aims at identifying zero-day attacks rather than known vulnerabilities. Network diversity is used to evaluate risk assessment, which is inspired by bio-diversity. Network diversity can be regarded as a kind of node metrics, which is the same as k-zero day safety. However, different from k-zero day safety, network diversity focuses on remote access resources.

It is a common belief that greater diversity in software and services may help to improve the network security [4]. Network diversity is modeled as a security metric for evaluating the robustness of networks against potential zero-day attacks [33].

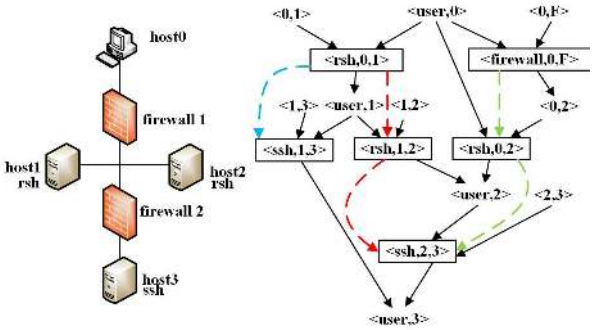


FIGURE 9. Network configuration and resource graph.

An example of network configuration and resource graph is given in Fig. 9 [33]. In Fig. 9, *host*0 is an attacker in the external network and aims to own the *user* privilege of *host*3, *host*1 and *host*2 provide *rsh* service, *host*3 provides *ssh* service, *firewall*1 allows the access to *host*1 but blocks to *host*2, *firewall*2 allows the access from *host*1 or *host*2 to *host*3. Fig. 9 shows three attack paths for attacker to obtain the *user* privilege of *host*3, which are described by dotted lines.

A network G consists of a set of hosts $H = \{h_1, h_2, \ldots, h_n\}$ and resource types $R = \{r_1, r_2, \ldots, r_m\}$. The equations used for computing metrics (d_1, d_2, d_3) [33] are given as follows:

$$r(G) = \frac{1}{\prod_1^n p_i^{p_i}}, \quad d_1 = \frac{r(G)}{t} \tag{23}$$

where $t = \sum_{i=1}^{n} |res(h_i)|$, and $p_j = \frac{|\{h_i : r_j \in res(h_i)\}|}{t}$ $(1 \leq i \leq n, 1 \leq j \leq m)$. $r(G)$ denotes the effective richness of resources in network, and d_1 computes the number of distinct resources inside a network and applies similarity-sensitive bio-diversity metric to take similarity between different resource types into consideration. For example, in Fig. 9, $H = \{host_0, host_1, host_2, host_3\}$ and $R = \{firewall, rsh, ssh\}$, $t = 2+2+2+1 = 7$, and $p_2 = \frac{2+2+2+0}{7}$. The way to calculate $\{p_1, p_3\}$ is the same as p_2. In this way, d_1 can be calculated.

$$d_2 = \frac{min_{q \in seq(c_g)} |R(q)|}{min_{q' \in seq(c_g)} |q'|} \tag{24}$$

where C represents the set of security conditions, c_g indicates the goal condition and $c_g \in C$, for each $c \in C$ and $q \in seq(c)$, $R(q)$ denotes $\{r : r \in R, r \textit{ appears in } q\}$, and $min(.)$ returns the minimum value in a set. This metric is based on the least attack effort required for compromising important resources, and considers the causal relationship between resources. For example, in Fig. 9, the red dotted line contains three exploits, but two exploits contain the same resource (*rsh*). So d_2 of this attack path is $\frac{2}{3}$.

$$d_3 = \frac{p'}{p} \tag{25}$$

where p represents the conditional probability of c_g being satisfied if all initial conditions are true, and p' denotes the minimum value of p when some edges are deleted from R_s

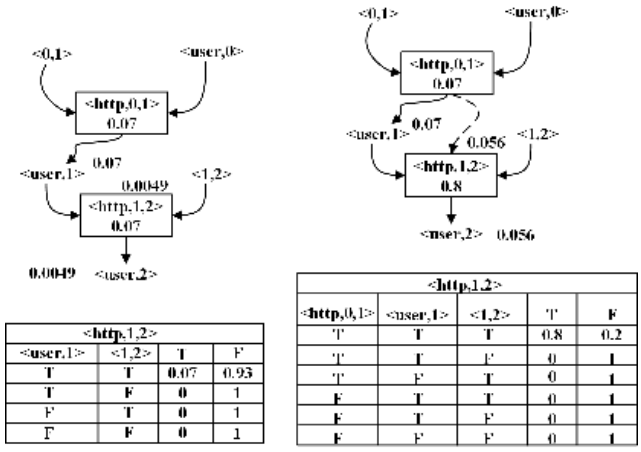


FIGURE 10. The impact of reusing an exploit.

(that is, edges from resources types to resource instances). This metric reflects the average attack effort required for compromising critical assets. For example, Fig. 10 shows the impact of reusing an exploit (*http*), which is described by the dotted lines on the right-hand side of Fig. 10. In Fig. 10, $d_3 = \frac{0.0049}{0.056}$.

Except for external factors, such as lack of data sets that can represent real network, the main limitations of their work are given as follows:

- Their proposed model relies on the availability and accuracy of inputs;
- Their work focuses on modeling diversity without considering the impact of other factors, such as the cost on maintenance;
- They assume that the probabilities of all assets on containing zero-day vulnerabilities are the same.

Based on this work, a new probabilistic model [36] is proposed for addressing the limitations of original d_3 metric. For example, invalid result will return during simulation, and once exploits are considered to be partially ordered, the attack likelihood will not necessarily be the lowest when all resources are assumed to be distinct [33]. The core idea of this new probabilistic model is to add a new parent node to exploits with the same resource type. An example of redesigned model is shown in Fig. 11. The left-hand side of Fig. 11 is from Fig. 10. The right-hand side of Fig. 11 represents the idea of resigning model for d_3. The limitation of their work is the high complexity of analyzing a resource graph.

In order to model different resources and the causal relationship among resources, the concept of resource graph is extended, and a labeled directed graph called extended resource graph is proposed [37]. The definition of extended resource graph is given as follows.

Extended Resource Graph [37]: Given a network composed of

- *a set of hosts H,*
- *a set of services S, with the service mapping serv*$(.) : H \rightarrow 2^S$,

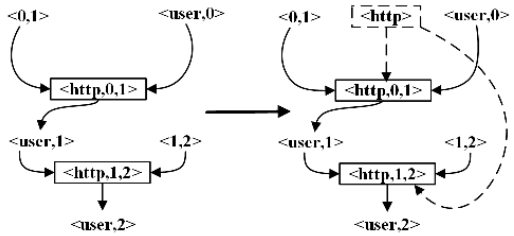


FIGURE 11. An example of redesigned model.

- *the collection of service pools* $SP = \{sp(s)|s \in S\}$,
- *and the labelling function* $v(.) : E \rightarrow SP$, *which satisfies* $\forall h_s \in S$, $\forall h'_s \in S$, $v(< s, h_s, h_d >) = v(< s, h'_s, h_d >)$ *(meaning all exploits with common service and destination host must be associated with the same service instance)*

Let E *be the set of zero-day exploits* $\{< s, h_s, h_d > | h_s \in H, h_d \in H, s \in serv(h_d)\}$, *and* $R_r \subseteq C \times E$ *and* $R_i \subseteq E \times C$ *be the collection of pre and post-conditions in* C. *The labeled directed graph* $< G(E \cup C, R_r \cup R_i), v >$ *is called the extended resource graph.*

In an extended resource graph, each pair represents a security-related condition, and each row below the rectangle indicates different hardening option, which is available for the condition. Each exploit node includes the information of a service running on the destination host and source host. The limitation of their work is that all service instances are assumed to have the same probability to be exploited. Therefore, they improve this work by considering the uneven distribution of services along an attack path [38].

2) DISCUSSION ON RG FROM ADVANTAGES, FLAWS AND SOLUTIONS

Resource Graph (RG) and traditional AG are closely related. The reason is that RG is syntactically equivalent to AG. For this reason, RG can be constructed by existing tools in Table 2, which are originally used to construct traditional AG.

However, the limitations of resource graph are the existence of cycles and high complexity of analyzing a resource graph, which are the same as traditional attack graph and zero-day attack graph. The solutions of these two problems can refer to the methods mentioned in attack graph. Meantime, this model only focuses on remote access resources (such as services or applications that are reachable from other hosts in the network), and the availability and accuracy of inputs are required during the process of forming a resource graph.

In the future, initial exploits of client-side applications, insider attacks and user mistakes should be considered when modeling resource graph.

C. BAYESIAN NETWORK (BN)

Bayesian Network (BN) is a Directed Acyclic Graph (DAG), which is also called belief network. Nodes represent random variables $\{x_1, x_2, \ldots, x_n\}$, and edges denote conditional independences between variables. Each node has a corresponding Conditional Probability Table (CPT), which is used to quantify the effect of the parent node on child node. Data source of nodes in BN comes from software vulnerabilities and open services. They belong to the application layer of OSI model.

The model constructed by applying the BN to attack graph is called Bayesian Attack Graph (BAG). Detailed analysis of this model will be carried out with an example. Meantime, the extended models of BN, including Dynamic Bayesian Networks (DBN), Bayesian Decision Network (BDN), and Fuzzy Probability Bayesian Network (FPBN) will also be analyzed in detail when introducing the development process of BN. Here the definition of BN is given below at first.

Bayesian Network [39]: Given a set of random variables $X = \{x_1, x_2, \ldots, x_n\}$ *in a Bayesian network, the joint probability of all variables is given by a chain rule with the following equation:*

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|Pa(x_i)), \tag{26}$$

where $Pa(x_i)$ *indicates that the specific value of the variable is in the parent node* x_i. *Bayesian network, also known as Bayesian belief network, is based on Bayes theory. The theorem equation is given as follows:*

$$P(X \mid Y) = \frac{P(X)P(Y \mid X)}{P(Y)}, \tag{27}$$

where $P(X \mid Y)$ *represents the posterior probability,* $P(X)$ *denotes the prior probability,* $P(Y \mid X)$ *means the probability that event* Y *occurs under the condition of event* X, *and* $P(Y)$ *represents the probability that event* Y *occurs without condition limitation.*

In literature [40], to provide a more compact representation of attack paths, the concept of Bayesian Attack Graph (BAG) is first proposed and applied to probabilistic analysis of risk assessment. However, the formal definition of BAG is not given, which can be found in [14]. Vulnerability risk assessment based on BAG is shown in Fig. 12. And the definition of BAG is given after Fig. 12. This model depends on two assumptions. First, given a node X_i, each parent node of X_i can independently influence the state of X_i. Second, once reaching a compromised state, an attacker will never need backtrack (that is, the assumption of monotonicity mentioned in attack graph). The local conditional probability distribution at node i is computed by (28):

$$p(x_i = 1|pa_i) = 1 - \prod_j (1 - p(x_i = 1|x_j)) \tag{28}$$

where $x_i = 1$ is a *true* state, which denotes a host with a compromised state, and pa_i indicates the parent node of x_i in BAG.

Bayesian Attack Graph [14]: Let S *be a set of attributes and* A *be the set of atomic attacks defined on* S. *A Bayesian Attack Graph is a tuple* $BAG = (S, \tau, \varepsilon, P)$, *where*

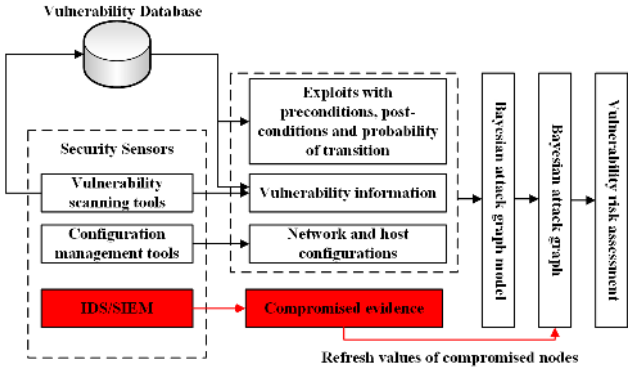


FIGURE 12. Vulnerability risk assessment based on BAG.

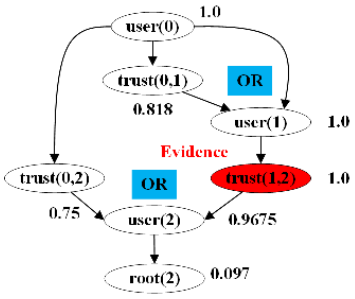


FIGURE 13. Unconditional probabilities for BAG with compromised evidence of node *trust*(1, 2).

- $S = N_{internal} \cup N_{external} \cup N_{terminal}$. $N_{external}$ *denotes the set of attributes* S_i *for which* $\nexists a \in A | S_i = post(a)$. $N_{internal}$ *denotes the set of attributes* S_j *for which* $\exists a_1, a_2 \in A | [S_j = pre(a_1)$ *and* $S_j = post(a_2)]$. $N_{terminal}$ *denotes the set of attributes* S_k *for which* $\nexists a \in A | S_k = pre(a)$.
- $\tau \subseteq S \times S$. *An ordered pair* $(S_{pre}, S_{post}) \in \tau$ *if* $S_{pre} \mapsto S_{post} \in A$. *Further, for* $S_i \in S$, *the set* $Pa[S_i] = S_j \in S | (S_j, S_i) \in \tau$ *is called the parent set of* S_i.
- ε *is a set of decomposition tuples of the form* $< S_j, d_j >$ *defined for all* $S_j \in N_{internal} \cup N_{terminal}$ *and* $d_j \in \{AND, OR\}$. d_j *is AND if* $S_j = 1 \Rightarrow \forall S_i \in Pa[S_j], S_i = 1$. d_j *is OR if* $S_j = 1 \Rightarrow \exists S_i \in Pa[S_j], S_i = 1$.
- P *is a set of discrete conditional probability distribution functions. Each attribute* $S_j \in N_{internal} \cup N_{terminal}$ *has a discrete local conditional probability distribution (LCPD) representing the values of* $Pr(S_j | Pa[S_j])$.

An example of applying Bayesian network to calculate the probability that an attacker can reach each state (condition) is shown in Fig. 13. To calculate the conditional probability distributions $p(X_i \mid pa_i)$ [41], which represent the probabilities that an attacker reaches security state X_i when given the observations of the set of preconditions pa_i. Two possible cases should be considered, including logical *AND* and logical *OR*, which are similar to the idea of cumulative score in attack graph. They belong to probabilistic metrics, and their calculation formulas are given respectively in (29) and (30):

$$p(X_i \mid pa_i) = \begin{cases} 0, & \exists X_j \in pa_i \mid X_j = F \\ \prod_{j:X_j} p_{v_j}, & \text{otherwise} \end{cases} \tag{29}$$

$$p(X_i \mid pa_i) = \begin{cases} 0, & \forall X_j \in pa_i \mid X_j = F \\ 1 - \prod_{j:X_j} (1 - p_{v_j}), & \text{otherwise} \end{cases} \tag{30}$$

where p_{v_j} denotes the probability that an attacker successfully exploits a vulnerability v_j.

For example, the probability of node *user*(2) in Fig. 13 belongs to the logical *OR* case. So the probability of this node is:

- $P(user(2)) = 1 - (1 - P(rsh(0, 2))(1 - P(rsh(1, 2)) = P(rsh(0, 2)) + P(rsh(1, 2)) - P(rsh(0, 2)) \times P(rsh(1, 2)) = 1 - (1 - 0.675)(1 - 0.9) = 0.675 + 0.9 - 0.675 \times 0.9 = 0.9675$

Different from [40], in literature [42], the probabilities of successful exploits are assigned to nodes rather than edges. The advantage of this method is that it can effectively combine the standard measurement such as CVSS to quantify the exploitability of each node. Moreover, in literature [42], Dynamic Bayesian Network (DBN), which can incorporate temporal factors into attack graph-based security metrics, is used to capture the evolving nature of vulnerabilities. In literature [43], DBN is also used to perform dynamic risk assessment. Their work has strong subjectivity because CPTs in their model are based on expert knowledge. Fuzzy comprehensive evaluation methods can be used to improve the objectivity of risk assessment. The definition of DBN is given below.

Dynamic Bayesian Network [44]: A Dynamic Bayesian Network (DBN) is an extended model of Bayesian Network and models probability distributions over semi-infinite collection of random variable variables, $Z_1, Z_2, \ldots$. *Variables are typically partition into* $Z_t = (U_t, X_t, Y_t)$ *to separately represent the input, hidden and output variables of a state-space model. The index t is increased by one every time a new observation arrived. The observation represents something has changed, which makes a model of a discrete-event system.*

DBN is defined to be a pair, $(B_1, B_{\rightarrow})$, *where* B_1 *is a BN which defines the prior* $P(Z_1)$, *and* $B_{\rightarrow}$ *is a two-slice temporal Bayesian network (2TBN) which defines* $P(Z_t | Z_{t-1}.)$ *by means of a Directed Acyclic Graph (DAG) as follows:*

$$P(Z_t | Z_{t-1}) = \prod_{i=1}^{N} P(Z_t^i | Pa(Z_t^i)) \tag{31}$$

where Z_t^i *is the ith node at time t, which could be a component of* X_t, Y_t *or* U_t, *and* $Pa(Z_t^i)$ *are the parents of* Z_t^i *in the graph. The nodes in the first slice of a 2TBN do not have any parameters associated with them, but each node in the second slice of the 2TBN has an associated conditional probability distribution (CPD), which defines* $P(Z_t^i | Pa(Z_t^i))$ *for all* $t > 1$.

In literature [45], Bayesian network is used to capture the uncertainty in attack structures, attacker actions and alerts. To capture the uncertainty in attack structure, CVSS metrics such as *Access Complexity* (*AC*) and *Exploitability* (*E*) are used to derive the CPT parameters.

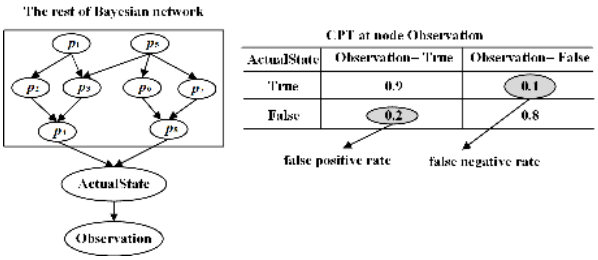


FIGURE 14. Local observation model.

TABLE 5. Inference algorithms.

Name	Type	Application
Belief propagation (BP)	Message passing	For attack trees
Junction tree (JT)	Message passing	For attack graphs
Variable elimination (VE)	Heuristic	For Bayesian networks
Forward-backward propagation	Heuristic	For chains
Most probable explanation (MPE)	Belief revision	Rely on VE

To capture the uncertainty in attacker actions, an Attack Action Node (AAN) is proposed. Attack action exists if the state of an AAN is *true*, otherwise, the probability of vulnerability exploit is 0. However, the correctness of evaluation depends on the reliability of security sensors.

A Local Observation Model (LOM) is used to model the uncertainty in alerts, which is shown in Fig. 14. More specifically, a pair of nodes, including *ActualState* node and *Observation* node, are introduced to Bayesian network. The *Observation* node is the direct child of the *ActualState* node. The *ActualState* node cannot observe itself, but can obtain an inference of its own state through the *Observation* node. If the *Observation* node obtains the *true* state based on evidence, the posterior probability of the *ActualState* node will be refreshed by computing $P(ActualState|Observation = True)$. This model relies on the reliability of evidence from security sensors. Common inference algorithms are summarized in Table 5.

In literature [46], Bayesian decision network is proposed to yield scalability, and integrate the risk assessment and outcome. The premise of using this model is to generate Bayesian attack graph of the network at first. CVSS metrics are used to calculate the exploitability of each single vulnerability, and the equation is given in (32).

$$Exploitability = 2 * AV * AC * AU \tag{32}$$

where AV denotes *Attack Vector*, AC indicates *Attack Complexity*, and AU represents *Authentication*. Time factors may have an impact on the probability of exploits. Therefore, besides the base metrics of CVSS (AV, AC and AU), the temporal metrics of CVSS, including *Exploitability* (E), *Remediation Level* (RL), and *Report Confidence* (RC) are also used in this work. The formula of *Temporal Probability* (TP) is given in (33).

$$TP = (E * RL * RC) * Exploitability \tag{33}$$

With the consideration of temporal metrics of CVSS, Equation (29) and (30) are changed to (34) and (35). The definition of Bayesian decision network is given after (35).

$$p(X_i \mid pa_i) = \begin{cases} 0, & \exists X_j \in pa_i \mid X_j = F \\ \prod_{j:X_j} TP_{v_j}, & \text{otherwise} \end{cases} \tag{34}$$

$$p(X_i \mid pa_i) = \begin{cases} 0, & \forall X_j \in pa_i \mid X_j = F \\ 1 - \prod_{j:X_j}(1 - TP_{v_j}), & \text{otherwise} \end{cases} \tag{35}$$

Bayesian Decision Network [46]: This model represents a decision network which combines a Bayesian network with additional node types for actions and utilities. Bayesian decision network contains three types of nodes. First, chance nodes, which denote random variables. Second, decision nodes, which indicate points where the decision maker has a choice of actions. Third, utility nodes, which represent the utility function of agent.

The utility node represents the expected utility (EU) associated with each action given the evidence as defined by

$$EU(A|E) = \prod_i P(O_i|E, A)U(O_i|A) \tag{36}$$

where E is the available evidence, A is an action with possible outcome states O_i, $U(O_i|A)$ is the utility of each of the outcome states, given that action A is taken, and $P(O_i|E, A)$ is the conditional probability distribution over the possible outcome states, given that evidence E is observed and action A is taken.

In addition to the applications on traditional network security, Bayesian network is also used in security risk assessment of Industrial Control Systems (ICSs). In literature [47], a novel model called Fuzzy Probability Bayesian Network (FPBN) is proposed to evaluate dynamic risk assessment of ICSs. To solve the problem of limited historical data, fuzzy probabilities (that is, the set of $\tilde{p}$ in the definition) are used in their model instead of crisp probabilities (that is, the set of p in the definition) in standard Bayesian network. Both expert knowledge and evidence are required for risk assessment in this model. To reduce the impact from noise evidences, a noise evidence filter is embedded in the inference algorithm. The definition of FPBN is given below.

Fuzzy Probability Bayesian Network [47]: Here BN is defined as $BN =< x, g^{x\mapsto x}, p >$, where

- *$x = (x_1, x_2, \ldots, x_{\ell(x)})$ is a set of $\ell(x)$ nodes in total.*
- *$g^{x\mapsto x}$ is an $\ell(x) \times \ell(x)$ incidence matrix that describes the relationship between the nodes, it is expressed as*

$$g^{x\mapsto x} = \begin{pmatrix} x_1 & x_2 & \cdots & x_{\ell(x)} \\ g_{1,1} & g_{1,2} & \cdots & g_{1,\ell(x)} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,\ell(x)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{\ell(x),1} & g_{\ell(x),2} & \cdots & g_{\ell(x),\ell(x)} \end{pmatrix} \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_{\ell(x)} \end{matrix} \tag{37}$$

TABLE 6. Development process of KVRA based on BN.

Literature	Year	Model	Network Type	Evidence and Expert Knowledge	Future Development
[40]	2005	BN	Traditional network	Evidence is required	Validation of effectiveness on enterprise-level networks.
[42]	2008	DBN	Traditional network	Not mentioned	Considering temporal metrics of CVSS and security hardening with the least cost.
[45]	2010	BN	Traditional network	Evidence is required	Verifying the scalability of the method in a large-scale real network.
[46]	2014	BDN	Traditional network	Evidence is required	Validation under the condition that only part of local hosts provide remote desktop service.
[43]	2016	DBN	Traditional network	Expert knowledge is required	Decreasing the subjectivity of risk assessment by fuzzy comprehensive evaluation.
[47]	2018	FPBN	Enterprise network	Both required	Decreasing the subjectivity of risk assessment by fuzzy comprehensive evaluation.

The definition of incidence matrix element $g_{i,j}$ is

$$g_{i,j} = \begin{cases} 1, & \text{node } x_i \text{ is the parent of node } m_j \\ 0, & \text{otherwise} \end{cases} \quad (38)$$

- $p = (p_1, p_2, \ldots, p_{\ell(x)})$ *is a set of conditional probability tables,* p_i *is the conditional probability table of node* x_i.

Fuzzy Probability Bayesian Network (FPBN) is defined as $FPBN =< x, g^{x \mapsto x}, \tilde{p}, v >$, *where*

- $x = (x_1, x_2, \ldots, x_{\ell(x)})$ *is a set of nodes,* x_i *represents an ICS event with three states T (true), F (false) and U (unknown):*

$$x_i = \begin{cases} T & \text{event of node } x_i \text{ happens} \\ F & \text{event of node } x_i \text{ does not happen} \\ U & \text{unknown} \end{cases} \quad (39)$$

There are four types of nodes in the FPBN: attack node a, function node f, incident node e, and asset node z. The event of an attack node means that an attacker launches an attack a. The event of a function node indicates that the system function f fails. The event of an incident node implies that a hazardous incident e happens. The event of an asset node marks a damage of the asset z.

- $g^{x \mapsto x}$ *is an* $\ell(x) \times \ell(x)$ *incidence matrix. It describes the relationship between the nodes, which is the same as standard BN.*
- $\tilde{p} = (\tilde{p_1}, \tilde{p_2}, \ldots, \tilde{p_{\ell(x)}})$ *is a set of conditional probability tables, and* $\tilde{p_i}$ *is the fuzzy conditional probability table of node* x_i.
- $v = (v_1, v_1, \ldots, v_{\ell(x)})$ *is a set of loss,* v_i *is the loss of node* x_i. *If* x_i *is an asset node, the loss* v_i *is the value of that asset; otherwise* $v_i = 0$. *There are three types of assets in ICSs: humans, environment, and properties.*

Development process of Known Vulnerability Risk Assessment (KVRA) based on Bayesian Network (BN) is summarized in Table 6. Although works mentioned above make a certain contribution to network security, they only focus on known vulnerabilities. Next, the way to apply BN to unknown vulnerability risk assessment and related works will be introduced.

1) APPLYING BN TO UVRA

In unknown vulnerability risk assessment, nodes in BN represent possible states, and edges denote state transitions. Each node has a Conditional Probability Table (CPT) that is used to quantify the effect of the parent node on child node.

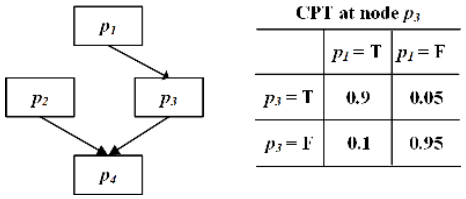


FIGURE 15. An example of Bayesian network.

The prior probability and conditional probability of nodes are subjective because they come from vulnerability database with manual evaluation. To make an objective assessment, evidence from security sensors can be used to update the probability of the compromised node. Meantime, the probabilities of nodes that have relationships (parent nodes and child nodes) will also be updated.

For example, in Fig. 15, if node p_1 is *true*, the probability of node p_3 being *true* is 0.9, and $P(p_3 = T|p_1 = T) = 0.9$. If evidences from security sensors such as IDSs confirm the fact that $p_3 = T$, the posterior probability of $P(p_1 = T|p_3 = T)$ can be calculated.

In literature [45], zero-day vulnerability is first mentioned in the domain of using Bayesian network for network security analysis, but the technology of unknown vulnerability risk assessment is not given.

In literature [48], Bayesian network is used to estimate the likelihood of acquiring critical software vulnerabilities and exploits. This model contains 13 states (nodes) and 17 activities (edges). Besides vulnerability databases, data of the activities in this Bayesian network contains previous empirical studies and a survey with 58 individuals who discover critical software vulnerabilities. The possibility of each state being true depends on the likelihood of its related activities (or steps) being true. Meantime, the latter relies on the characteristics of software and resources of an attacker.

The advantage of this model is that it can be used to support enterprise decision making. The limitations of this model mainly include three aspects. First, this work only focuses on exploiting the root/administrator permissions of the victim host through remote access. Second, this model only considers software programs required to be compiled. Third, skills and resources of attackers will impact the decision result.

An example of the model proposed in [48] is shown in Fig. 16. The number in this figure indicates an activity (edge), and the letter indicates a state (node). Equation (40)

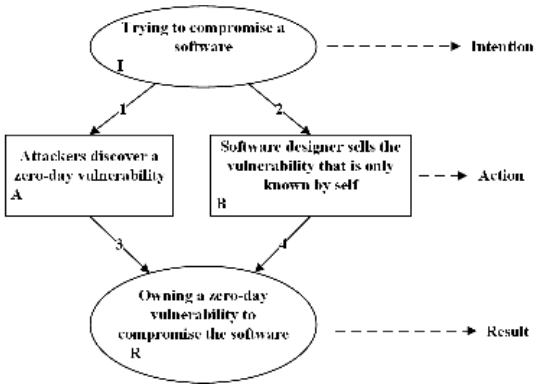


FIGURE 16. An example of BN for software exploit.

and (41) are required to calculate P_1 and P_2.

$$P_1 = P(X \leq T_w | X \in LI(2, 9, 855)) \tag{40}$$

where T_w denotes work days for discovering a zero-day vulnerability, LI indicates linear interpolation.

$$P_2 = P(X \leq M | X \in EXP(1.32 * 10^{-5})) \tag{41}$$

where M represents money (United States dollar) for purchasing a zero-day vulnerability, and EXP indicates exponential distribution. Equation (42) and (43) respectively denote the formula of linear interpolation and exponential distribution.

$$LI(X) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(X - x_0) \tag{42}$$

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{43}$$

In literature [48], when $T_w = \{2, 9, 855\}$, the likelihood of discovering a zero-day vulnerability $P_D = \{5\%, 50\%, 95\%\}$, and $P_3 = P_4 = 1$. If $T_w = 50 \in [9, 855]$, $M = 7000$, and $\lambda = 1.32 * 10^{-5}$. The calculation process of vulnerability risk assessment in Fig. 16 is given as follows:

- $P_1 = 50\% + \frac{95\%-50\%}{855-9} * (50 - 9) = 52.2\%$
- $P_2 = 1.32 * 10^{-5} e^{-1.32*10^{-5}*7000} = 8.8\%$
- $P_A = P_1 = 52.2\%, P_B = P_2 = 8.8\%$
- $P_R = P_A P_3 \cup P_B P_4 = 52.2\% * 1 + 8.8\% * 1 - 52.2\% * 1 * 8.8\% * 1 = 56.4\%$

If $P_A P_3$ and $P_B P_4$ are regarded as S_1 and S_2 respectively, which can be founded in the definition of attack graph, it can be clearly discovered that the idea of the cumulative score in attack graph is applied to calculating P_R. The future work of this model is to consider other types of vulnerabilities.

To investigate the possibility of improving the *tolerance* for Industrial Control Systems (ICSs) with zero-day attack by defending against known weakness, in literature [34], a model called Bayesian Risk Network (BRN) is proposed. In this model, nodes represent different meanings, including target nodes, attack nodes and requirement nodes. Edges represent correlation between nodes. Data source of nodes in BRN comes from the ICS Top 10 Threats and Countermeasures, and Common Cybersecurity Vulnerabilities in ICSs. They belong to the application layer of OSI model.

To measure the minimum effort required for zero-day exploits to compromise a system, *tolerance* is defined as a probabilistic metric. Then, Bayesian network is used to analyze the zero-day threat propagation across ICSs. Attackers can choose a known or zero-day weakness at each step to propagate the risk. If the exploitability of the chosen weakness and its previous exploited target are obtained, the probability of successful exploit can be computed. The definition of Bayesian Risk Network is given below.

Bayesian Risk Network [34]: Let $B =< N, P_T, P_E, P_R, P_{T_0} >$ *be a Bayesian Risk Network, where*

- $N = T \cup \varepsilon \cup R$, *including target nodes, attack nodes and requirement nodes.*
- $P_T = \{P_{T_1}, \ldots, P_{T_n}\}$ *includes conditional probabilities of all non-root target nodes given their parents such that* P_{T_x} *denotes* $P(T_x | \bigcup_{T'_x \in pa(T_x)} E_{T'_x T_x})$, *where* $P(T_x | \bigcup_{T'_x \in pa(T_x)} E_{T'_x T_x}) = 1 - \prod_{T'_x \in pa(T_x)} (1 - P(T_x | E_{T'_x T_x}))$ *by noisy-OR operator.* $P(T_x | E_{T'_x T_x})$ *is the probability of* T_x *given the weakness used at* $E_{T'_x T_x}$.
- $P_E = \{P_{E_{T'_1 T_1}}, \ldots, P_{E_{T'_n T_n}}\}$ *includes conditional probability distribution for all attack nodes such that* $P_{E_{T'_x T_x}}$ *denotes* $P(E_{T'_x T_x} | T'_x)$.
- $P_R = \{P_{R_1}, \ldots, P_{R_n}\}$ *includes decomposition of all requirement nodes such that* P_{R_x} *denotes* $P(R_x | pa(R_x))$, *where* $P(R_x | pa(R_x)) = \sum_{R'_x \in pa(R_x)} P(R_x | R'_x)$, *and* $P(R_x | R'_x)$ *is the assigned proportion of* R'_x *in* R_x.
- P_{T_0} *is the prior probability distribution of the root node* T_0.
- $P(T_x)$ *is the unconditional probability of* $T_x \in T$, *which can be obtained by:*

$$P(T_x) = \begin{cases} \sum_{E_{T'_x T_x}} P_{T_x} \sum_{T'_x} P_{E_{T'_x T_x}} P(T'_x), & if\ \omega_z \notin \Omega(E_{T'_x T_x}) \\ \sum_{E_{T'_x T_x}} P_{T_x} \sum_{T'_x} P_{E_{T'_x T_x}} P(T'_x) & \\ + P(T_x | E_{T'_x T_x} = \omega_z) & \\ \sum_{T'_x} P_{E_{T'_x T_x}} P(T'_x), & otherwise \end{cases} \tag{44}$$

  $P(T_x)$ *is obtained by its parent node* $P(T'_x)$ *recursively until it hits the root* T_0 *whose probability distribution is known.* $\sum_{E_{T'_x T_x}}$ *denotes* $E_{T'_x T_x}$ *is marginalized.* P_{T_x}, P_{R_x} *and* $P_{E_{T'_x T_x}}$ *are given by* P_T, P_R *and* P_E *respectively.* $P(T_x | E_{T'_x T_x} = \omega_z)$ *equals to the uncertain exploitability of the zero-day exploit* ω_z *at* T_x.
- $P(R_x)$ *denotes the unconditional probability of* R_x *inR given its parents* R'_x *and* $P(R_x) = \sum_{R'_x} P_{R_x} \prod_{R'_x \in pa(R_x)} P(R'_x)$, *where* $pa(R_x)$ *are marginally independent.*

An example of Bayesian risk network with no control deployed is given in Fig. 17 [34]. White ovals denote target nodes, grey ovals indicate attack nodes, and blue ovals represent requirement nodes. The exploitability of weakness $\{w1, w2, w3, w4, w5\}$ is $\{80\%, 60\%, 70\%, 80\%, 60\%\}$. w means the weight of corresponding target node.

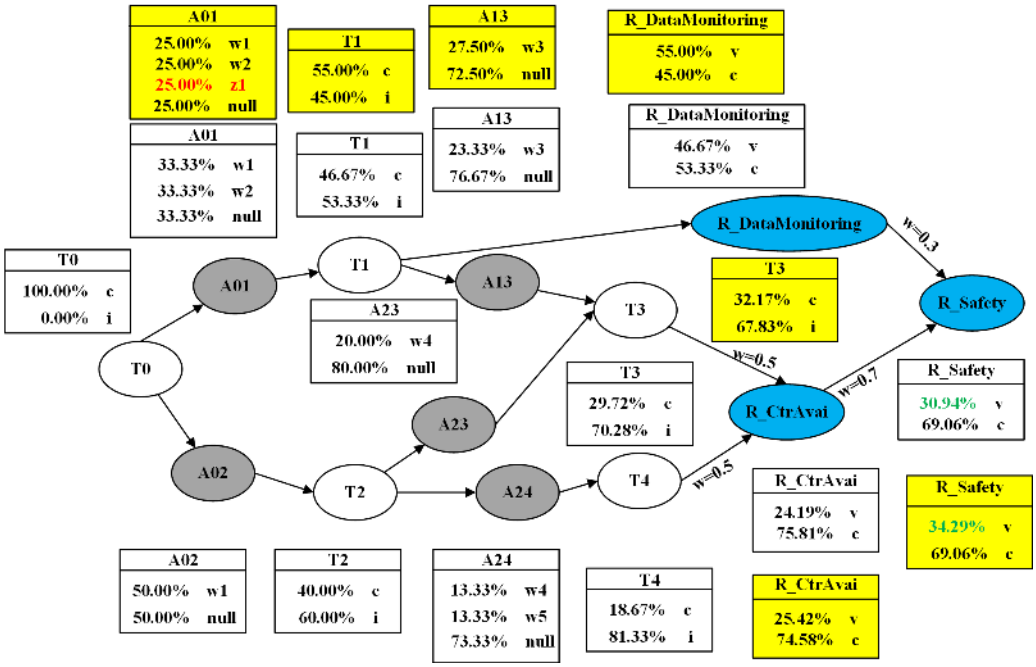


FIGURE 17. Bayesian risk network with no control deployed.

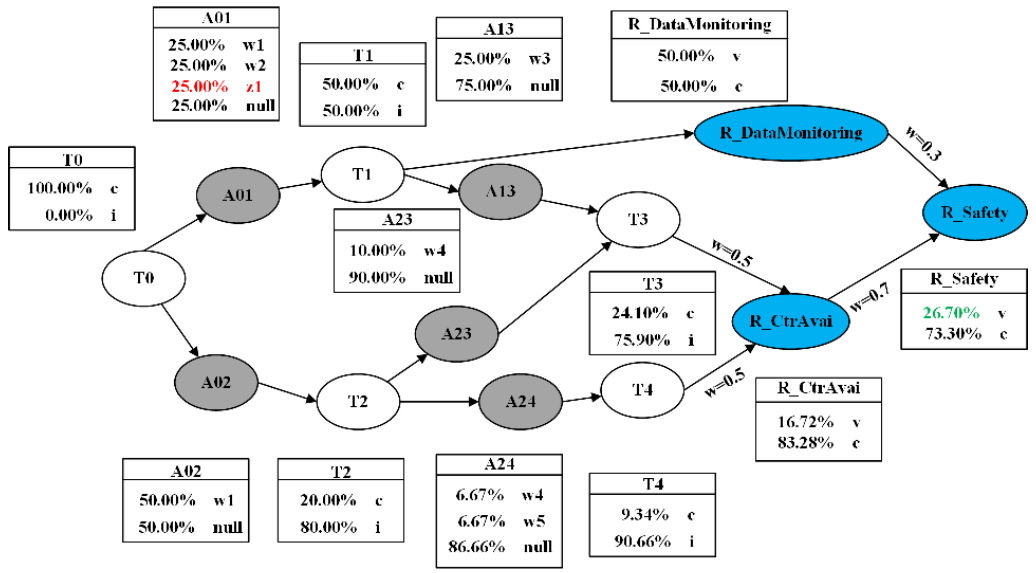


FIGURE 18. Bayesian risk network with control deployed.

There are two cases in Fig. 17. Conditional Probability Tables (CPTs) with white color denote the case with no zero-day exploit, and the risk is 30.94%. CPTs with yellow color indicate the case that zero-day exploit $z1$ exists in the network, and the exploitability is set to 80%. Under this circumstance, the risk rises from 30.94% to 34.29%.

Compared with Fig. 17, in Fig. 18, the exploitability of $z1$ is set to 100%. And control is deployed to combat $w1$, which decreases half exploitability of $w1$ (from 80% to 40%). Under this circumstance, the risk decreases from 34.29% to 26.70%, which is also lower than the first case in Fig. 17. In other words, with the implementation of reasonable control, the *tolerance* of the system against zero-day exploits can be improved.

Although *tolerance* and k-zero day safety are different, they are similar. More specifically, they are both looking for a value to indicate the condition that the systems (or other network assets) can be compromised. It is worth noting that the idea of calculating logical *OR* node in Bayesian attack graph is applied to calculating the compromised probability

of target node $T3$. The future development of this graph model is to cancel the assumption that the exploitability of each weakness will always decrease 50% under control deployed.

2) DISCUSSION ON BN FROM ADVANTAGES, FLAWS AND SOLUTIONS

The advantages of Bayesian network mainly include three aspects. First, Bayesian network provides a causal-consequence relationship between random variables, which is similar to attack graph. Second, Bayesian network is a directed acyclic graph and provides a more compact representation with all attack path information. In other words, Bayesian network shows better scalability in medium and large networks. Third, Bayesian network provides a form of reasoning partial belief under uncertain conditions.

However, Bayesian network also exists flaws. For example, Bayesian network includes boundary constraints on probability of state values of the variables. To solve this problem, fuzzy probabilities can be used to replace the crisp probabilities in standard Bayesian networks. Meantime, it is important to investigate whether the BN models used for problems associated with insiders are applicable for Industrial Control System (ICS) environments, especially for a control room with an operator [34].

Bayesian risk network is an extended model of Bayesian network, which can be used to improve the safety of Industrial Control Systems (ICSs). This model is constructed at the level of assets rather than states and attributes. The advantage of BRN is that it can model zero-day exploits with the limitation of details about them (such as pre-requisites or post-conditions) [34].

However, there are three types of nodes in this model, and the probability calculation methods are different according to the node type. That is, the complexity of analyzing a Bayesian risk network is high. Moreover, this work only considers individual zero-day weakness at different targets. Meantime, it does not consider the cost of deploying control solution [8]. To further verify the validity of this model, the case that one attack path contains multiple zero-day exploits needs to be considered. In addition, the cost (mainly includes human resource, time, and money) of each solution should also be given.

Looking forward to future development, attacks from insiders also need to be considered to verify whether this model can address both internal and external issues as well as social engineering attacks, collusion attacks, etc.

IV. DGM FOR UVRA BASED ON SYSTEM-LEVEL DATA

This section introduces directed graph models for unknown vulnerability risk assessment whose nodes are constructed by system-level data. Due to the lack of related works, only the system call is discussed in this section. Directed graph models that rely on system-level data include System Object Dependency Graph (SODG) and Object Instance Graph (OIG).

The rest of this section is arranged as follows. First, the concepts of these models are given, including the definitions and purposes. Next, the general process of applying these directed graphs to UVRA is given, and related works of these models will be analyzed. Finally, these directed graph models will be discussed from three aspects, including advantages, flaws and solutions.

A. SYSTEM OBJECT DEPENDENCY GRAPH (SODG)

In literature [49], a model called System Object Dependency Graph (SOGD) is proposed to reveal zero-day attack paths. Nodes in SODG represent system objects, such as files, processes and sockets. Edges in SODG indicate the dependency between system calls. Labels on edges represent when the system calls occur. The definition of SODG is given below.

System Object Dependency Graph [49]: If the system call trace for the $i-th$ host is denoted as $\sum_i$, then the SODG for the host is a directed graph $G(V_i, E_i)$, where:

- *V_i is the set of nodes, and initialized to empty set $\emptyset$;*
- *E_i is the set of directed edges, and initialized to empty set $\emptyset$;*
- *If a system call $syscall \in \sum_i$, and dep is the dependency relation parsed from syscall, where $dep \in \{(src \rightarrow sink), (sink \rightarrow src), (src \leftrightarrow sink)\}$, src and sink are OS objects (mainly a process, file or socket), then $V_i = V_i \cup \{src, sink\}$, $E_i = E_i \cup \{dep\}$. dep inherits timestamps start and end from syscall;*
- *If $(a \rightarrow b) \in E_i$ and $(b \rightarrow c) \in E_i$, then c transitively depends on a.*

1) APPLYING SODG TO UVRA

To build a complete SODG of a managed network, each host should construct its own SODG at first. System calls are parsed to generate nodes and edges in each SODG. The auditing of system calls on each host is performed at first to filter the useless information. Then, system call traces from individual hosts are sent to the analysis machine after filtering. Next, to obtain the complete SODG, different SODGs should be connected together if and only if there is at least one same directed edge that appears in different graphs simultaneously. Since it is difficult to discover zero-day vulnerabilities alone, zero-day attacks are identified by calculating the risk probability of Suspicious Intrusion Propagation Paths (SIPPs) [49]. SIPPs can be divided into path metric, and the definition of SIPPs is given below.

Suspicious Intrusion Propagation Paths [49]: If the network-wide SODG is denoted as $\cup G(V_i, E_i)$, where $G(V_i, E_i)$ denotes the per-host SODG for the $i-th$ host, then the SIPPs are s subgraph of $\cup G(V_i, E_i)$, denoted as $G(V', E')$, where:

- *V' is the set of nodes, and $V' \subset \cup V_i$;*
- *E' is the set of directed edges, and $E' \subset \cup E_i$;*
- *V' is initialized to include trigger nodes only. And trigger nodes come from SODG objects that are involved in the alert from security sensors (such as IDSs);*
- *For $\forall obj' \in V'$, if $\exists obj \in \cup V_i$ where $(obj \rightarrow obj') \in \cup E_i$ and $start(obj \rightarrow obj') \leq lat(obj')$, then $V' = V' \cup \{obj\}$*

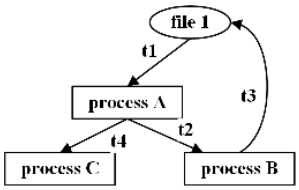


FIGURE 19. An example of SODG.

and $E' = E' \cup \{(obj \rightarrow obj')\}$. $lat(obj')$ *maintains the latest access time to* obj' *by edges in* E';

- *For* $\forall obj' \in V'$, *if* $\exists obj \in \cup V_i$ *where* $(obj \rightarrow obj') \in \cup E_i$ *and* $end(obj' \rightarrow obj) \geq eat(obj')$, *then* $V' = V' \cup \{obj\}$ *and* $E' = E' \cup \{(obj' \rightarrow obj)\}$. $eat(obj')$ *maintains the earliest access time to* obj' *by edges in* E'.

The use of SODG for UVRA appears in [49] at first. This work exists two main flaws. First, as shown in Fig. 19, SODG may exist cycles. Second, this work assumes that both pre-knowledge and common features at OS-level can be obtained, which is hard to accomplish in reality. The existence of cycles will lead to the complexity of risk assessment.

2) DISCUSSION ON SODG FROM ADVANTAGES, FLAWS AND SOLUTIONS

The advantage of this model is that it identifies zero-day attacks by paths rather than single exploit. The reason is that individual zero-day exploit is hard to detect in reality.

The flaw of this graph model is the existence of cycles, which will lead to the difficulty of quantification. Meantime, the readability of SODG will become worse with the growth of network scale.

A model called object instance graph is proposed to solve the problems of SODG. This model will be introduced next.

B. OBJECT INSTANCE GRAPH (OIG)

To solve the problem of SODG that it may exist cycles, in literature [50], a model called object instance graph is proposed. Each node in object instance graph indicates an object instance, and each edge represents the dependency relation between nodes. The definition of object instance graph is given below.

Object Instance Graph [50]: If the system call trace in a time window $T[t_{begin}, t_{end}]$ *is denoted as* $\sum_T$ *and the set of system objects (mainly processes, files or sockets) involved in* $\sum_T$ *is denoted as* O_T, *then the object instance graph is a directed graph* $G_T(V, E)$, *where:*

- V *is the set of nodes, and initialized to empty set* $\emptyset$;
- E *is the set of directed edges, and initialized to empty set* $\emptyset$;
- *If a system call* **syscall** $\in \sum_T$ *is parsed into two system object instances* src_i, $sink_j$, $i, j \geq 1$, *and a dependency relation* dep_c: $src_i \rightarrow sink_j$, *where* src_i *is the* i^{th} *instance of system object* $src \in O_T$, *and* $sink_j$ *is the* j^{th} *instance of system object* $sink \in O_T$, *then* $V = V \cup \{src_i, sink_j\}$, $E = E \cup \{dep_c\}$. *The timestamps for syscall,* dep_c, src_i, *and* $sink_j$ *are respectively denoted as* $t\_syscall$, t_dep_c, t_src_i, *and* t_sink_j. *The* t_dep_c *inherits* $t\_syscall$ *from syscall. The index i and j are determined before adding* src_i *and* $sink_j$ *into V by:*
  - *For* $\forall src_m, sink_n \in V$, $m, n \geq 1$, *if* i_{max} *and* j_{max} *are respectively the maximum indexes of instances for object src and sink, and;*
  - *If* $\exists src_k \in V$, $k \geq 1$, *then* $i = i_{max}$, *and* t_src_i *stays the same; Otherwise,* $i = 1$, *and* t_src_i *is updated to* $t\_syscall$;
  - *If* $\exists sink_z \in V$, $z \geq 1$, *then* $j = j_{max} + 1$; *Otherwise,* $j = 1$. *In both cases* t_sink_j *is updated to* $t\_syscall$; *If* $j \geq 2$, *then* $E = E \cup \{dep_s : sink_{j-1} \rightarrow sink_j\}$.
- *If* $a \rightarrow b \in E$ *and* $b \rightarrow c \in E$, *then c transitively depends on a.*

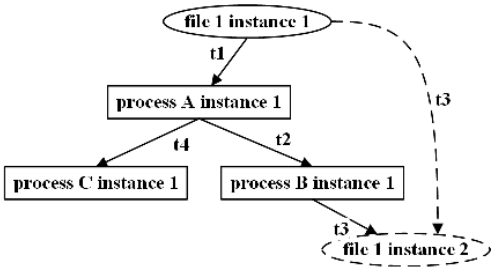
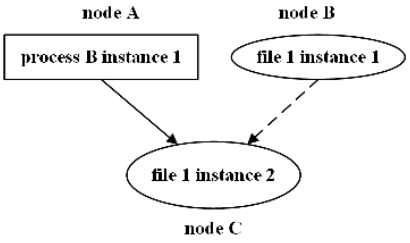


FIGURE 20. An example of object instance graph.



CPT at node C

	B = Infected		B = Uninfected	
	A = Infected	A = Uninfected	A = Infected	A = Uninfected
C = Infected	1	1	τ	ρ
C = Uninfected	0	0	$1-\tau$	$1-\rho$

FIGURE 21. The infection propagation model.

1) APPLYING OIG TO UVRA

Object instance graph relies on SODG. The process of applying SODG to UVRA can be founded in the previous section. Object instance graph eliminates the cycles in SODG by creating corresponding number of object instances according to the frequency that an object being called.

An example of object instance graph is given in Fig. 20. From Fig. 20, it can be clearly discovered that the cycle exists in Fig. 19 is eliminated. The infection propagation model is shown in Fig. 21, which contains two different cases.

First, when there are instances from different timestamps of the same object (*file* 1), if an instance of the previous

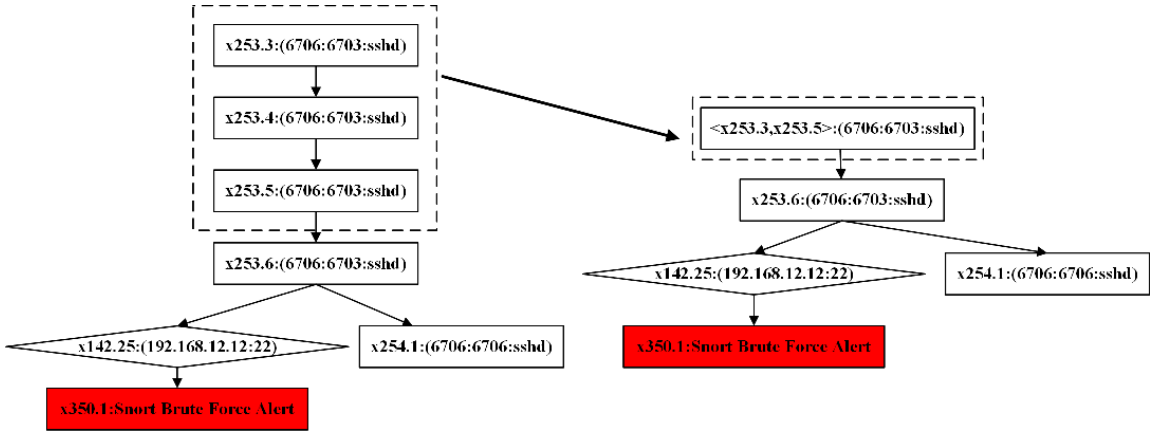


FIGURE 22. Merging different instance nodes of the same object.

timestamp (*node B*) is infected, the instance of the latter timestamp (*node C*) must also be infected.

Second, if an instance of the previous timestamp (*node B*) is not infected, the instance of the latter timestamp (*node C*) may be infected by other object (*node A*). Contact infection rate τ denotes the probability that *node C* gets infected when *node A* is infected, and intrinsic infection rate ρ indicates the probability that *node C* gets infected when *node A* is not infected.

To eliminate the assumption that all pre-knowledge and common features at OS-level can be obtained in [49], Bayesian network is applied to object instance graph because it can model cause-and-effect relations, and incorporate intrusion evidences from various security sensors. In this work, Local Observation Model (LOM) is used to incorporate evidences, which has been introduced in Fig. 14. The future development of this graph model is to solve the problem that some attack paths cannot be revealed when attack time span exceeds the analysis time period.

2) DISCUSSION ON OIG FROM ADVANTAGES, FLAWS AND SOLUTIONS

The advantage of this model is that Bayesian network can be applied to this model for calculating the probability of nodes being infected.

Scalability is one of the defects of this model, because multiple object instances will be generated if system objects are called frequently. To solve the problem of scalability, besides the pruning operations already mentioned in [50], another way is to merge different instance nodes of the same object. These instance nodes do not include evidences from security sensors, and nodes whose compromised probabilities are lower than the threshold should also be excluded. Meantime, these nodes that can be merged only have one child node. After merging, the representation is $< t_{start}, t_{end} >: (pid : ppid : pcmd)$, where t_{start} denotes the start time of merging, t_{end} indicates the end time of merging, *pid* means process ID, *ppid* represents the ID of parent process, and *pcmd* is process command. An example of this idea is given in Fig. 22. Nodes in the dotted line on the left-hand side of Fig. 22 are required to be merged. The node in the dotted box on the right-hand side of Fig. 22 is the merging result. As can be seen from Fig. 22, this idea simplifies the graph on the basis of retaining the information in the original version.

In addition, when some attack activities evade system calls, or the attack time span exceeds the analysis time period, the constructed instance graphs may not capture the complete zero-day attack paths. Genetic algorithm can be used to solve this problem because it is highly suited to NP-complete problems, such as searching through all attack paths [51].

V. CHALLENGES AND SOLUTIONS

The technology of Unknown Vulnerability Risk Assessment (UVRA) based on directed graph models is still in the development stage, and many problems are urgently required to be solved. Meantime, the common challenges of mentioned directed graph models for UVRA can be summarized into following aspects:

- Obtaining zero-day vulnerabilities. Zero-day vulnerabilities are difficult to obtain, current researches often set a time point to divide known vulnerabilities into known and unknown vulnerabilities [8], [21]. For example, setting the time point to 2018/12/31, vulnerabilities before 2018/12/31 are used as known vulnerabilities, and vulnerabilities occur after this time point are regarded as zero-day vulnerabilities;
- Generating directed graphs effectively. At present, the directed graphs given in the existing research results are based on a managed network with a small number of hosts. With the growth of network scale, the complexity of attack graph generation will increase;
- Proposing better security metrics. The application scope of current security metrics is limited. Security metrics for Unknown Vulnerability Risk Assessment (UVRA) based on Directed Graph Models (DGMs) are categorized in Table 7, where CVSS denotes Common

Vulnerability Scoring System, CCSS indicates Common Configuration Scoring System, CWSS represents Common Weakness Scoring System, SP indicates Shortest Path, and SIPPs denotes Suspicious Intrusion Propagation Paths. Better security metrics are required to obtain more accurate exploitation, which depends on comprehensive considerations of vulnerabilities, assets, etc.;

- The improvement of directed graph models and risk assessment methods. The applications of existing models have certain limitations. Further improvement on the theoretical system of unknown vulnerability risk assessment based on directed graphs is required.

Combining existing researches and prospect for future development, possible solutions for the above challenges are given as follows:

- Due to the difficulty of obtaining zero-day vulnerabilities, the method that dividing them from known vulnerabilities by setting a time point will be continuously used in the future. Another possible approach is when dividing zero-day vulnerabilities from known vulnerabilities, the time for fixing a vulnerability should also be considered as a classification indicator;
- The growth of network nodes inevitably increases the time complexity for generating attack graphs. One solution is to use the idea of distributed attack graph generation [52] to overcome the state space explosion problems with the growth of hosts and vulnerabilities in a managed network;
- Not only the inherent characteristic of the vulnerability (exploitability), but also time and environmental factors of security metrics should be considered to obtain a more accurate risk assessment. When multiple security metrics are used to quantify network security risks, fuzzy comprehensive risk assessment can be used to determine the severity of the risk. The successful application of bio-diversity in risk assessment reflects the possibility of integration between different scientific research fields. Some methods regard alerts from security sensors as evidences to dynamically update the exploitability of corresponding nodes, so the accuracy of risk assessment can be indirectly enhanced by improving the reliability of security sensors;
- To obtain a better model, fusion between directed graph models is an existing method (such as Bayesian attack graph which is the fusion product of attack graph and Bayesian network) and ongoing approach.

VI. DISCUSSION

Unknown Vulnerability Risk Assessment (UVRA) focuses on zero-day vulnerabilities. A typical zero-day attack may last for 310 days on average [53]. To protect against zero-day attacks, various methods are proposed. These methods are classified as statistics, signatures and behavior techniques [54]. In order to meet the future trend, the classification needs to be further expanded. The following content introduces the future work directions of UVRA from the perspective of techniques and application trends.

A. MAIN TECHNIQUES FOR URVA

The main techniques for UVRA are listed as follows, including current popular techniques and methods that will play an important role in the future. Their defects at the current stage will be analyzed, which can be regarded as the future work directions of UVRA.

1) STATISTICAL-BASED TECHNIQUES

Zero-day detection based on statistical techniques relies on static attack profile and manual modifications of detection settings. For example, Nessus is a vulnerability scanner, which is often used in vulnerability risk assessment. The principle of Nessus is to compare the defects of hosts with the vulnerability features stored in the vulnerability database. The list of hosts that need to be scanned is manually configured by users. If the defects match the vulnerability features, they will be regarded as vulnerabilities. Therefore, this method is not suitable for real-time detection and defense.

2) SIGNATURE-BASED TECHNIQUES

As mentioned before, suspicious activities whose signatures are not defined previously in Snort IDS/IPS are regarded as zero-day exploits. A signature is a sequence of bytes at specific locations within the executable, a regular expression, a hash value of binary data, or any other formats created by malware analyst which should accurately identify malware instances [55]. This kind of method relies on previous malware signatures, which means unknown malware without known signatures will not be detected.

3) BEHAVIOR-BASED TECHNIQUES

Methods based on behaviors assume that malware can be detected by observing the malicious behaviors exhibited by malware during runtime [55]. Compared with signature-based techniques, behavior-based methods focus on observing malware actions instead of previously known signatures. Therefore, behavior-based methods have a better performance on malware variants with similar behaviors but different structures. However, their flaws are the high false positive rate and worse performance on mimicry attacks.

4) GRAPH THEORY

This method is the core content of our paper. It can be clearly discovered that the advantage of this method is to visualize the network status by graphs. Each path represents the strategy that attackers may use to achieve the goal. However, the probability that each attack path being adopted by attackers cannot be known by only using the knowledge of graph theory. Among the current researches of UVRA based on directed graph models, most of them use both directed graphs and security metrics to complete qualitative and quantitative tasks.

TABLE 7. The classification of security metrics for UVRA based on DGMs.

Metric Sets	Metric Subsets	Individual Metrics
Standard metrics	/	CVSS, CCSS, CWSS
Specific metrics	Node metrics	k-zero day safety, network diversity (d_1, d_2, d_3)
	Path metrics	SP, SIPP, length of the k-zero day safety
	Probabilistic metrics	Cumulative score, logical *And*, logical *OR*, intrinsic infection rate (ρ), contact infection rate (τ), *tolerance*

5) SECURITY METRICS

In UVRA, security metrics are used to quantitatively evaluate the probability that network systems will be exploited when facing attacks. Security metrics are often used in conjunction with graph models, which is the main theme of our paper. The previous classification work of security metrics can be seen in Table 4. However, there is a gap in the previous work to classify the security metrics for UVRA. Therefore, the supplementary work is made in our paper, which can be seen in Table 7.

6) PROBABILITY THEORY

The original purpose of proposing probability theory is to analyze the frequency of events. These events can be repeated, such as throwing a coin and observing whether the coin falls to the front or the back. However, some events cannot be repeated. For example, supposing the probability of a host being exploited is p, and assuming the threshold to be reached for an emergency fix is q. If $p > q$, the host requires to be repaired immediately. Otherwise, it does not need to do that. The former is called the frequentist probability, which is directly related to the frequency of events. The latter is called the Bayesian probability, which involves the level of certainty. Bayesian theory enables us to infer the uncertainty of network attacks. The combination of Bayesian theory and graph theory is Bayesian network.

7) ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) can be simply regarded as using computer science to simulate human thinking. It has been called one of the three cutting-edge technologies (genetic engineering, nanoscience, and artificial intelligence) of the world in the 20th century. And it is also considered to be one of the cutting-edge technologies (space, energy, and artificial intelligence) of the world in the 21th century.

AI will play a vital role in vulnerability risk assessment. Methods for vulnerability risk assessment can be divided into three steps [56].

The first step is manual vulnerability assessment. This method is time costing and depends on expert knowledge deeply.

The second step is assistive vulnerability assessment, which is the stage of most current researches. This method requires the help of vulnerability scanners or frameworks to find the most relevant security weakness. Therefore, the lack of flexibility and compatibility will be the inherent defects of this method. And the dependency on expert knowledge still exists.

The third step is fully automated vulnerability assessment, which is based on the technology of artificial intelligence. Compared with the first and second step, this method can produce expert-level decisions without the help of human beings, which will reduce the costs on time and economic. However, this technology is still in the development stage, and further researches on automated vulnerability mitigation techniques that can actually protect computing platforms are still required [56].

B. APPLICATION TRENDS OF UVRA

Unknown Vulnerability Risk Assessment (UVRA) focuses on traditional network at first. However, devices connected to the network are no longer just hosts, they show a trend of diversification. The result is that existing UVRA techniques may lose their power on these devices, such as intelligent robots, vehicles, etc. Therefore, it is important to correctly grasp the application trends of UVRA. Due to the variety of application scenarios, the following content only focuses on the future work directions of UVRA on Industrial Internet of Things (IIoT), robots and vehicles.

1) IIoT

Nowadays, the era of Internet of Things (IoT) is coming. The number and types of devices connected to IoT are far beyond traditional networks. In other words, there are more vulnerabilities in IoT than traditional networks. Moreover, with a constant growth of attack surfaces and capabilities, network systems will suffer from more serious attacks. Therefore, UVRA is a key technology to keep IoT security. Industrial IoT (IIoT) is an application of IoT on industry, with the aim of improving the overall operational efficiency in industrial management [57]. Events that exploiting the vulnerabilities in the devices of IIoT happen frequently recent years. The future work directions on applying UVRA to IIoT are as follows:

- New security metrics to quantify vulnerabilities in IIoT. The cost and return on patching a vulnerability should be considered;
- New assessment frameworks and specific vulnerability database for IIoT. Due to the difficulty to update the underlying light weight operating system, many vulnerabilities in IoT devices cannot be expected to be patched and upgraded. Therefore, new assessment frameworks

should be able to provide risk mitigation strategies to improve security;
- New risk vectors that satisfy the form of International IoT Asset Classification (IIoTAC) and Key IoT Cyber Risk Factors (KIoTCRF) [58];
- New methods to solve the low attack path quantification degree and complex path finding.

2) ROBOTS

With the rise of AI technology, different types of intelligent robots are constantly being connected to the network. However, robots are typically not created with security as a main concern [59]. Therefore, the existence of vulnerabilities in robots is an inevitable fact. At present, researches on the vulnerability risk assessment of robots are lacking. In order to achieve an effective vulnerability risk assessment, the following points can be the future work directions:
- New security metrics to quantify vulnerabilities in robots. A novel security metric called Robot Vulnerability Scoring System (RVSS) [59] is proposed to evaluate the severity caused by corresponding vulnerabilities. This metric is an improvement of CVSS. However, there is almost no relevant metric for zero-day vulnerabilities in robots;
- New assessment framework and specific vulnerability database for robots. Due to the difference between robot and conventional vulnerabilities on features, some reported robot vulnerabilities are difficult to be classified by existing standards. Moreover, the severity assessment of robot vulnerabilities is also quite different from conventional vulnerabilities.

3) VEHICLES

Besides robots, vehicles also connect to the network and become network/physical systems, which are the core component of Intelligent Transportation System (ITS). Vulnerabilities in vehicles can cause the loss of property, even the personal safety of users. However, the research on assessing the impact of network/physical attacks on road users and system operators is lacking [60]. Although literature [60] takes a step forward in the vulnerability risk assessment of vehicles, it only finishes the qualitative work of vulnerabilities. The future work directions on vulnerability risk assessment of vehicles are listed as follow:
- Trying to apply Bayesian network to in-vehicle network. The reason is that there is a high degree of uncertainty about the impact of security breaches on the in-vehicle network [60]. Bayesian network can infer uncertainty and visualize attack paths;
- Proposing novel security metrics to quantify vulnerabilities in vehicles. A common idea is to improve existing standards to adapt new application scenarios.

VII. CONCLUSION

Directed graph models for unknown vulnerability risk assessment are formed by enumerating possible attack paths (or called attack strategies) from the perspective of attackers. They play an important role on risk assessment and decision making for administrators. In this paper, the concepts and definitions of directed graph models are given at first. Then, these models are analyzed from three aspects, including advantages, flaws and solutions. Next, corresponding examples are given to facilitate understanding. After that, challenges and solutions of unknown vulnerability risk assessment based on directed graph models are given. Meantime, Security metrics for unknown vulnerability risk assessment based on directed graph models are summarized and classified. At last, future work directions of UVRA are discussed from the perspective of techniques and applications. At present, the survey of unknown vulnerability risk assessment based on directed graph models is relatively lacking, so our work is valuable.

REFERENCES

[1] C. M. Alina, S. E. Cerasela, and G. Gabriela, "Cyber attacks and combat behavior," *Ovidius Univ. Ann.*, vol. 17, no. 1, pp. 141–144, Jun. 2017.

[2] K. Durkota, V. Lisý, C. Kiekintveld, B. Bošanský, and M. P chou ek, "Case studies of network defense with attack graph games," *IEEE Intell. Syst.*, vol. 31, no. 5, pp. 24–30, Sep./Oct. 2016, doi: 10.1109/MIS.2016.74.

[3] C. Wu, T. Wen, and Y. Zhang, "A revised CVSS-based system to improve the dispersion of vulnerability risk scores," *Sci. China Inf. Sci.*, vol. 62, Mar. 2019, Art. no. 39102, doi: 10.1007/s11432-017-9445-4.

[4] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "*k*-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 1, pp. 30–44, Jan./Feb. 2013, doi: 10.1109/TDSC.2013.24.

[5] L. Wang, S. Jajodia, A. Singhal, and S. Noel, "*k*-zero day safety: Measuring the security risk of networks against unknown attacks," in *Proc. Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2010, pp. 573–587.

[6] J. Srinivas, A. K. Das, and N. Kumar, "Government regulations in cyber security: Framework, standards and recommendations," *Future Gener. Comput. Syst.*, vol. 92, pp. 178–188, Mar. 2019, doi: 10.1016/j.future.2018.09.063.

[7] E. Zio, "The future of risk assessment," *Rel. Eng. Syst. Saf.*, vol. 177, pp. 176–190, Sep. 2018, doi: 10.1016/j.ress.2018.04.020.

[8] X. Sun, J. Dai, P. Liu, and A. Singhal, and J. Yen, "Using Bayesian networks for probabilistic identification of zero-day attack paths," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2506–2521, Oct. 2018, doi: 10.1109/TIFS.2018.2821095.

[9] M. Keramati, "An attack graph based procedure for risk estimation of zero-day attacks," in *Proc. 8th Int. Symp. Telecommun. (IST)*, Tehran, Iran, Sep. 2016, pp. 723–728, doi: 10.1109/ISTEL.2016.7881918.

[10] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *Proc. IFIP Annu. Conf. Data Appl. Secur.*, 2008, pp. 283–296, doi: 10.1007/978-3-540-70567-3_22.

[11] L. P. Swiler and C. Phillips, "A graph-based system for network-vulnerability analysis," in *Proc. Workshop New Secur. Paradigms*, 1998, pp. 71–79, doi: 10.1145/310889.310919.

[12] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proc. 9th ACM Conf. Comput. Commun. Secur. (CCS)*, 2002, pp. 217–224, doi: 10.1145/586110.586140.

[13] M. Frigault and L. Wang, "Measuring network security using Bayesian network-based attack graphs," in *Proc. 32nd Annu. IEEE Int. Comput. Softw. Appl. Conf.*, Jul./Aug. 2008, pp. 698–703, doi: 10.1109/COMPSAC.2008.88.

[14] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 1, pp. 61–74, Jan. 2012, doi: 10.1109/TDSC.2011.34.

[15] S.-C. Liu and Y. Liu, "Network security risk assessment method based on HMM and attack graph model," in *Proc. 17th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Shanghai, China, May/Jun. 2016, pp. 517–522, doi: 10.1109/SNPD.2016.7515951.

[16] F. Dai, Y. Hu, K. Zheng, and B. Wu, "Exploring risk flow attack graph for security risk assessment," *IET Inf. Secur.*, vol. 9, no. 6, pp. 344–353, Nov. 2015, doi: 10.1049/iet-ifs.2014.0272.
[17] H. Wang, Z. Chen, J. Zhao, X. Di, and D. Liu, "A vulnerability assessment method in industrial Internet of Things based on attack graph and maximum flow," *IEEE Access*, vol. 6, no. 1, pp. 8599–8609, 2018, doi: 10.1109/ACCESS.2018.2805690.
[18] H. A. Kholidy, A. Erradi, S. Abdelwahed, and A. Azab, "A finite state hidden Markov model for predicting multistage attacks in cloud systems," in *Proc. IEEE 12th Int. Conf. Dependable, Autonomic Secure Comput.*, Dalian, China, Aug. 2014, pp. 14–19, doi: 10.1109/DASC.2014.12.
[19] L. Liu, J. Xu, C. Guo, J. Kang, S. Xu, and B. Zhang, "Exposing SQL injection vulnerability through penetration test based on finite state machine," in *Proc. 2nd IEEE Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Oct. 2016, pp. 1171–1175, doi: 10.1109/CompComm.2016.7924889.
[20] M. Albanese, S. Jajodia, A. Singhal, and L. Wang, "An efficient framework for evaluating the risk of zero-day vulnerabilities," in *Proc. Int. Conf. E-Bus. Telecommun. (ICETE)*, 2013, pp. 322–340, doi: 10.1007/978-3-662-44788-8_19.
[21] C. Joshi, U. K. Singh, and D. Kanellopoulos, "An enhanced framework for identification and risks assessment of zero-day vulnerabilities," *Int. J. Appl. Eng. Res.*, vol. 13, no. 12, pp. 10861–10870 Jul. 2018.
[22] S. Noel, S. Jajodia, L. Wang, and A. Singhal, "Measuring security risk of networks using attack graphs," *Int. J. Next-Gener. Comput.*, vol. 1, no. 1, pp. 135–147, Jul. 2010.
[23] M. Frigault, L. Wang, S. Jajodia, and A. Singhal, "Measuring the overall network security by combining CVSS scores based on attack graphs and Bayesian networks," in *Network Security Metrics*. Cham, Switzerland: Springer, Nov. 2017, pp. 1–23, doi: 10.1007/978-3-319-66505-4_1.
[24] J. Sembiring, M. Ramadhan, Y. S. Gondokaryono, and A. A. Arman, "Network security risk analysis using improved MulVAL Bayesian attack graphs," *Int. J. Elect. Eng. Inform.*, vol. 7, no. 4, pp. 735–753, Dec. 2015, doi: 10.15676/ijeei.2015.7.4.15.
[25] V. Mavroeidis and S. Bromander, "Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence," in *Proc. Eur. Intell. Secur. Inform. Conf.*, Athens, Greece, Sep. 2017, pp. 91–98, doi: 10.1109/EISIC.2017.20.
[26] N. Gao, Y. He, and B. Ling, "Exploring attack graphs for security risk assessment: A probabilistic approach," *Wuhan Univ. J. Natural Sci.*, vol. 23, no. 2, pp. 171–177, Apr. 2018, doi: 10.1007/s11859-018-1307-0.
[27] A. Singhal and X. Ou, "Security risk analysis of enterprise networks using probabilistic attack graphs," in *Network Security Metrics*. Cham, Switzerland: Springer, Nov. 2017, pp. 53–73, doi: 10.1007/978-3-319-66505-4_3.
[28] S. Noel and S. Jajodia, "Metrics suite for network attack graph analytics," in *Proc. 9th Cyber Inf. Secur. Res. Conf.*, 2014, pp. 5–8, doi: 10.1145/2602087.2602117.
[29] N. Idika and B. Bhargava, "Extending attack graph-based security metrics and aggregating their application," *IEEE Trans. Dependable Secure Comput.*, vol. 9, no. 1, pp. 75–85, Jan./Feb. 2012, doi: 10.1109/TDSC.2010.61.
[30] A. Ramos, M. Lazar, R. H. Filho, and J. J. P. C. Rodrigues, "Model-based quantitative network security metrics: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2704–2734, 4th Quart., 2017, doi: 10.1109/COMST.2017.2745505.
[31] S. E. Yusuf, J. B. Hong, M. Ge, and D. S. Kim, "Composite metrics for network security analysis," *Softw. Netw.*, vol. 2017, no. 1, pp. 137–160, Jan. 2018, doi: 10.13052/jsn2445-9739.2017.007.
[32] K. Prasad, S. Kumar, A. Negi, and A. Mahanti, "Generation and risk analysis of network attack graph," in *Proc. 4th Int. Conf. Frontiers Intell. Comput., Theory Appl. (FICTA)*, 2015, pp. 507–516, doi: 10.1007/978-81-322-2695-6_42.
[33] L. Wang, M. Zhang, S. Jajodia, A. Singhal, and M. Albanese, "Modeling network diversity for evaluating the robustness of networks against zero-day attacks," in *Proc. Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2014, pp. 494–511.
[34] T. Li and C. Hankin, "Effective defence against zero-day exploits using Bayesian networks," in *Proc. Int. Conf. Crit. Inf. Infrastruct. Secur. (CRITIS)*, 2016, pp. 123–136, doi: 10.1007/978-3-319-71368-7_11.
[35] S. Kumar, A. Negi, K. Prasad, and A. Mahanti, "Evaluation of network risk using attack graph based security metrics," in *Proc. IEEE 14th Int. Conf. Dependable, Autonomic Secure Comput., 14th Int. Conf. Pervasive Intell. Comput., 2nd Int. Conf. Big Data Intell. Comput. Cyber. Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Auckland, New Zealand, Aug. 2016, pp. 91–93, doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2016.30.
[36] M. Zhang, L. Wang, S. Jajodia, A. Singhal, and M. Albanese, "Network diversity: A security metric for evaluating the resilience of networks against zero-day attacks," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 5, pp. 1071–1086, May 2016, doi: 10.1109/TIFS.2016.2516916.
[37] D. Borbor, L. Wang, S. Jajodia, and A. Singhal, "Diversifying network services under cost constraints for better resilience against unknown attacks," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2016, pp. 295–312, doi: 10.1007/978-3-319-41483-6_21.
[38] D. Borbor, L. Wang, S. Jajodia, and A. Singhal, "Securing networks against unpatchable and unknown vulnerabilities using heterogeneous hardening options," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2017, pp. 509–528, doi: 10.1007/978-3-319-61176-1_28.
[39] A. A. Ramaki, M. Khosravi-Farmad, and A. G. Bafghi, "Real time alert correlation and prediction using Bayesian networks," in *Proc. 12th Int. Iranian Soc. Cryptol. Conf. Inf. Secur. Cryptol. (ISCISC)*, Rasht, Iran, Sep. 2015, pp. 98–103, doi: 10.1109/ISCISC.2015.7387905.
[40] Y. Liu and H. Man, "Network vulnerability assessment using Bayesian networks," *Proc. SPIE*, vol. 5812, pp. 61–71, Mar. 2005, doi: 10.1117/12.604240.
[41] L. Muñoz-González, D. Sgandurra, M. Barrère, and E. C. Lupu, "Exact inference techniques for the analysis of Bayesian attack graphs," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 2, pp. 231–244, Mar./Apr. 2017, doi: 10.1109/TDSC.2016.2627033.
[42] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic Bayesian network," in *Proc. 4th ACM Workshop Qual. Protection*, 2008, pp. 23–30, doi: 10.1145/1456362.1456368.
[43] J. Wang, K. Fan, W. Mo, and D. Xu, "A method for information security risk assessment based on the dynamic Bayesian network," in *Proc. Int. Conf. Netw. Netw. Appl. (NaNA)*, Hakodate, Japan, Jul. 2016, pp. 279–283, doi: 10.1109/NaNA.2016.50.
[44] J. Brulé, "The computational power of dynamic Bayesian networks," *CoRR*, vol. 1603, pp. 158–166, Mar. 2016.
[45] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using Bayesian networks for cyber security analysis," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Chicago, IL, USA, Jun./Jul. 2010, pp. 211–220, doi: 10.1109/DSN.2010.5544924.
[46] M. Khosravi-Farmad, R. Rezaee, A. Harati, and A. G. Bafghi, "Network security risk mitigation using Bayesian decision networks," in *Proc. 4th Int. Conf. Comput. Knowl. Eng. (ICCKE)*, Mashhad, Iran, Oct. 2014, pp. 267–272, doi: 10.1109/ICCKE.2014.6993444.
[47] Q. Zhang, C. Zhou, Y.-C. Tian, N. Xiong, Y. Qin, and B. Hu, "A fuzzy probability Bayesian network approach for dynamic cybersecurity risk assessment in industrial control systems," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2497–2506, Jun. 2018, doi: 10.1109/TII.2017.2768998.
[48] H. Holm, M. Korman, and M. Ekstedt, "A Bayesian network model for likelihood estimations of acquirement of critical software vulnerabilities and exploits," *Inf. Softw. Technol.*, vol. 58, pp. 304–318, Feb. 2015, doi: 10.1016/j.infsof.2014.07.001.
[49] J. Dai, X. Sun, and P. Liu, "Patrol: Revealing zero-day attack paths through network-wide system object dependencies," in *Proc. Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2013, pp. 536–555, doi: 10.1007/978-3-642-40203-6_30.
[50] X. Sun, J. Dai, P. Liu, A. Singhal, and J. Yen, "Towards probabilistic identification of zero-day attack paths," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Philadelphia, PA, USA, Oct. 2016, pp. 64–72, doi: 10.1109/CNS.2016.7860471.
[51] M. Alhomidi and M. Reed, "Risk assessment and analysis through population-based attack graph modelling," in *Proc. World Congr. Internet Secur. (WorldCIS)*, London, U.K., Dec. 2013, pp. 19–24, doi: 10.1109/WorldCIS.2013.6751011.
[52] K. Kaynar and F. Sivrikaya, "Distributed attack graph generation," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 5, pp. 519–532, Sep./Oct. 2016, doi: 10.1109/TDSC.2015.2423682.
[53] "Zero-day danger: A survey of zero-day attacks and what they say about the traditional security model," FireEye Secur. Reimagined, White Paper, 2015.
[54] U. K. Singh and C. Joshi, "Scalable approach towards discovery of unknown vulnerabilities," *Int. J. Netw. Secur.*, vol. 20, no. 5, pp. 827–835, Sep. 2018, doi: 10.6633/IJNS.201809_20(5).03.
[55] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *J. Comput. Virol. Hacking Techn.*, vol. 12, no. 2, pp. 59–67, May 2016, doi: 10.1007/s11416-015-0244-0.

[56] S. Khan and S. Parkinson, "Review into state of the art of vulnerability assessment using artificial intelligence," in *Guide to Vulnerability Analysis for Computer Networks and Systems*, Sep. 2018, pp. 3–32, doi: 10.1007/978-3-319-92624-7_1.

[57] G. George and S. M. Thampi, "A graph-based security framework for securing industrial IoT networks from vulnerability exploitations," *IEEE Access*, vol. 6, pp. 43586–43601, Aug. 2018, doi: 10.1109/ACCESS.2018.2863244.

[58] P. Radanliev, D. C. De Roure, R. Nicolescu, M. Huth, R. M. Montalvo, S. Cannady, and P. Burnap, "Future developments in cyber risk assessment for the Internet of Things," *Comput. Ind.*, vol. 102, pp. 14–22, Nov. 2018, doi: 10.1016/j.compind.2018.08.002.

[59] V. M. Vilches, E. Gil-Uriarte, I. Z. Ugarte, G. O. Mendia, R. I. Pisón, L. A. Kirschgens, A. B. Calvo, A. H. Cordero, and L. Apa, C. Cerrudo, "Towards an open standard for assessing the severity of robot security vulnerabilities, the robot vulnerability scoring system (RVSS)," *Robot Cybersecur.*, vol. 1807, pp. 1–25, Jul. 2018.

[60] K. B. Kelarestaghi, M. Foruhandeh, K. Heaslip, and R. Gerdes, "Vehicle security: Risk assessment in transportation," *CoRR*, vol. 1804, pp. 1–6, Apr. 2018.

WENHAO HE received the B.S. degree in network engineering from the Zijin College, Nanjing University of Science and Technology, in 2017. He is currently pursuing the master's degree with the College of Computer Science and Technology, Shanghai University of Electric Power, Shanghai, China. His research interests include network security risk assessment and intrusion detection.

HONGJIAO LI received the master's degree from the Huazhong University of Science and Technology, in 2002, and the Ph.D. degree from Shanghai Jiaotong University, in 2008. She is currently an Associate Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. Her main research interests include big data security and privacy, network security, and AI security.

JINGUO LI received the B.S. degree in information security and the Ph.D. degree in computer science and technology from Hunan University, China, in 2007 and 2014, respectively. He is currently an Associate Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. His research activities are focused on network security and privacy, and applied cryptography.