ILLATION: Improving Vulnerability Risk Prioritization By Learning From Network

Zhen Zeng, *Member, IEEE,* Dijiang Huang, *Senior Member, IEEE,* Guoliang Xue, *Fellow, IEEE,* Yuli Deng, Neha Vadnere, and Liguang Xie, *Senior Member, IEEE*

Abstract—Network administrators face the challenge of efficiently patching overwhelming volumes of vulnerabilities with limited time and resources. To address this issue, they must prioritize vulnerabilities based on the associated risk/severity measurements (i.e., CVSS). Existing solutions struggle to efficiently patch thousands of vulnerabilities on a network. This paper presents ILLATION, a proof-of-concept model that provides network-specific vulnerability risk prioritization to support efficient patching. ILLATION integrates AI techniques, such as neural networks and logical programming, to learn risk patterns from adversaries, vulnerability severity, and the network environment. It provides an integrated solution that learns and infers adversaries' motivation and ability in a network while also learning the constraints that restrict interactions between vulnerabilities and network elements. An evaluation of ILLATION against CVSS base and environmental metrics shows that it reflects changes in vulnerability scores and prioritization ranks as the same pattern as the CVSS model while identifying vulnerabilities with similar risk patterns to given adversaries better. On a simulated network with up to 10k vulnerable hosts and vulnerabilities, ILLATION can assess 1k vulnerabilities in about 4.5 minutes total, with an average running time of 0.19 seconds per vulnerability on a general-purpose computer.

Index Terms—Vulnerability Prioritization, Logic Reasoning, Network Security, Cloud Security.

1 Introduction

Vulnerability prioritization is an essential task for security remediation. It supports organizations in patching vulnerabilities efficiently and effectively [1], [2]. Vulnerability prioritization generates a prioritized list of vulnerabilities, which is used for managing and mitigating vulnerabilities. Routinely patching vulnerabilities on an enterprise network is a challenging task due to the overwhelming volume of vulnerabilities that outpace the available remediation resources [2]. According to the records on CVEDetails, the number of new vulnerabilities per year increases by over 70% from 2017 to 2022 [3]. The total amount of known vulnerabilities is over 196k in 2023 [3], and new vulnerabilities emerge at an average rate of 50 per day recently [4]. The traditional "one for all" vulnerability prioritization strategies are facing challenges in patching vulnerability efficiently with limited remediation resources, and the organization's objective-specific vulnerability prioritization strategy is highlighted in a recent report [5]. To address this challenge, we propose a network-specific vulnerability prioritization method in this paper.

The vulnerability risk is assessed as $(probability) \times (consequence)$ [2], [6], where the probability refers to vulnerability exploitation likelihood and the consequence refers to the consequence (impact) of the exploitation w.r.t. a cyber asset [2]. The predominant vulnerability prioritization method Common Vulnerability Scoring System (CVSS) also assesses a vulnerability from the perspectives of exploitability and impact in its Base Metrics by aggregating the values of expert-defined measurements [7]–[10]. All known vulnerabilities in the National Vulnerability Database (NVD) have CVSS scores calculated by the CVSS Base Metrics. These scores are widely used to prioritize vulnerability in practice [11], [12]. However, CVSS base scores do not correlate well with vulnerability exploits in practice [6], since adversaries' motivations and abilities are not explicitly considered in CVSS base scores [2]. Adversaries deliver a successful attack by exploiting vulnerabilities in a network. In general, adversaries' motivation and ability would determine which vulnerabilities in this network will be exploited by adversaries. From this perspective, adversaries' motivation and ability should be considered as one of the key risk features in vulnerability risk assessment [2], [13].

Prior research work also reveals that vulnerability risk correlates with network elements [7], [10], [14], where a vulnerability affects network security via complex interactions among vulnerabilities and network environments [14]–[16]. However, there is always *a trade-off between the accuracy of analyzing interactions in detail and the problem of information overload*. For example, CVSS provides Environmental metrics to adjust the requirements w.r.t. a network environment [7]–[10], where the categorical value of metrics requires configurations for each vulnerability. The attack graph models represent interactions as a series of exploits via symbolic modeling and logic programming [14]–[19]. Each attack graph indicates the abstracted network topology with a directed acyclic graph, which supports network defenders to understand critical threats and select countermeasures [20]. Information overload is one of the key limitations when implementing attack graphs to patch vulnerability

• *Authors were with the College of Engineering & Applied Science, University of Wisconsin Milwaukee, Milwaukee WI 53211, the School of Computing and Augmented Intelligence, Arizona State University, Tempe AZ 85281, and the Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg VA 24061.*

E-mail: zhenzeng@uwm.edu, {dijiang.huang, xue, ydeng19, nvadnere}@asu.edu, xie@vt.edu

in practice [21], [22]. Additionally, rule-based attack graph models can only generate results as per rules, which have less learning ability in attack mode [23].

We propose ILLATION, a network-specific vulnerability prioritization method, which addresses the challenge of vulnerability prioritization by providing an integrated solution that explicitly

1) learns adversaries' behavior in terms of motivation and ability in a network, and
2) tunes the initial risk assessment score with the constraints of interactions between vulnerability and network elements.

At a high level, ILLATION provides a *network-specific vulnerability risk prioritization to support efficient patches*. ILLATION assesses vulnerability risk from three main factors: the likelihood of exploit, the criticality of exploit, and the constraints that restrict interactions between vulnerability and network elements. The AI techniques of natural language processing (NLP), neuro-symbolic computation, and probabilistic logic programming are utilized to assess vulnerability risk. The key ideas and their significance are discussed below, and a detailed technical description is shown in Section 3.

First, ILLATION aims to assess vulnerability risk *from both the likelihood of exploit and criticality of exploit by considering the known or suspected adversaries' risk patterns in the given network*. In (Stage (a)), ILLATION learns the risk pattern of the adversary's motivation and ability from historical security data. By doing so, ILLATION can utilize such learned knowledge to assess the risk of new vulnerabilities. The second important idea in ILLATION is to *tune the initial risk assessment with the constraints of interactions between vulnerability and network elements.* Being different from attack graphs, ILLATION *analyzes the effects of network-vulnerability interactions by focusing on learning how interaction constraints restrict vulnerability exploits (in terms of the criticality of exploits and the likelihood of exploits) on the given network*. In (Stage (b)), ILLATION derives two interaction prerequisites from network profiles. A network profile can be viewed as an attack scenario of a network (i.e., what defensive system an adversary needs to penetrate for successful exploitation), and contains the configurations of the network and security setup (i.e., network topology, reachability, access control, firewall rules, security group, etc.). In this paper, ILLATION considers *vulnerable services* and *reachable hosts* as two interaction prerequisites. The success of a vulnerability exploit is prior impossible if no vulnerable service is running on a host machine [24]. Meanwhile, reachability is one of the fundamental prerequisites for modeling network attacks [20]. For example, for a successful client-side attack, a vulnerable client must have a network path to reach the malicious server. For a successful server-side attack, an attacker must reach the victim host's vulnerable server. To infer interaction constraints from these prerequisites, ILLATION considers which vulnerable service is running, how critical the service is assigned by cyber admins, which vulnerabilities are associated with this service, and to which degree a vulnerable service can affect other hosts on a network.

ILLATION is evaluated on both the case study and simulated networks. The case study [1] has a 5-node Kubernetes [25] cluster hosted in AWS EC2, and the simulated networks environment where have vulnerable hosts and vulnerabilities up to 10k. Compared with CVSS base and environmental metrics, ILLATION not only reflects changes in vulnerability scores and prioritization ranks as the same pattern as the CVSS model but also better identifies vulnerabilities with a similar risk pattern of the given adversaries. Additionally, ILLATION's prioritization reasoning engine can assess 1k vulnerabilities in the network with 10k hosts in about 4.5 minutes total. The average running time of assessing 10k vulnerabilities is up to 0.19 seconds per vulnerability in the scalable network on a general-purpose computer.

The rest of the paper is as follows. Section 2 describes the related work of vulnerability prioritization solutions. Section 3 provides descriptions of the proposed system and model. Section 4 evaluates ILLATION's reasoning engine. Section 5 discusses the limitations and future work. Section 6 concludes this paper. The appendix lists the related materials.

2 Related Work

In practice, defenders assess and prioritize the identified vulnerabilities based on the associated risk levels. It is very challenging to deal with the numerous discovered vulnerabilities in a timely fashion [26], [27] since a vulnerability scan tool can discover thousands or more vulnerabilities in a network [2] and new vulnerabilities emerge at an average rate of 50 per day recently [4]. Therefore, for a cybersecurity analyst, to understand which vulnerabilities are risky in a network [14] and to decide how to efficiently and effectively reduce the associated risks is critical.

According to the Verizon 2021 data breach investigations report, less than 25% of discovered vulnerabilities can be remediated (i.e., fixed or patched) within 30 days [1]. There can be several reasons that delay remediation in reality: (a) for a high-ranked vulnerability, the system may not have the capability (such as tools, financial support, or knowledge background of the defender) to remediate it based on a software upgrading/patching solution [28]; (b) defenders may want to postpone remediation due to management issues, such as the regular schedule of software release/upgrade, to avoid downtime that is required to patch a vulnerability [28]. We group the existing studies that support vulnerability prioritization in a network system into two categories and compare them with ILLATION.

Network Agnostic Approaches: The existing predominant vulnerability prioritization approach is the CVSS scoring schema [7]–[10]. The CVSS is based on a static formula and cannot automatically associate vulnerabilities with network environmental elements [29], [30]. Using the CVSS temporal and environmental metrics requires a security admin to manually adjust measurements for each vulnerability. Thus, it is extremely challenging to apply these two metrics in practice [31], [32]. The previous study [33] improves the implementation of these two metrics by estimating some missing data points in the CVSS environmental

1. A live demo of ILLATION in case study is running on the online platform https://dtai.cs.kuleuven.be/problog/editor.html#task=prob&hash=8b7de28aaa1856109f24ab57adcdf5b1

and temporal metrics. However, this approach is still based on the static metric formula and lacks the learning ability of the real vulnerability exploits. Additionally, in reality, the high CVSS scores do not correlate well with vulnerability exploits [6], [34]. To better learn vulnerability exploits in the real scenario, previous works use the AI techniques of machine learning [2], [13], [35]–[37]. However, vulnerability exploit records in the wild have attributes for assessing the likelihood of exploitation in general but have very limited attributes to indicate the security impact of vulnerability to a network [2], [6]. For example, a recent study [38] uses an ML-based model by only considering vulnerability location, protection level, and CVSS value, which fails to cover the interactions among vulnerabilities and network elements. Additionally, for the method of machine learning, the selection of training and testing datasets critically affects the performance of the ML-based model [39]. To overcome the limitations, ILLATION learns the risk pattern of vulnerability exploits in terms of the adversary's motivation and ability by training the assessment model.

Network Specific Approaches: The attack graph-based approaches [15], [17]–[19] capture interactions among vulnerabilities and network elements in detail by enumerating all possible attack paths to a given attacking target [15]. However, it is impractical to analyze real attack scenarios due to information overload [22], [23]. Because enumerating all possible attack "start" points and "target" points as [14], [19] would be very difficult in a large-scale network with hundreds or thousands of machines. For example, [40] has exponential complexity with network states since it defines an attack as a sequence of state transitions among atomic attack nodes within a network, and [14] analyzes vulnerability based on the limited exploitable range and consequence (only refers to privilege escalation), and cannot reason on vulnerabilities with confidentiality or integrity loss. Previous work [41] improves the attack graph generation to the scales nearly linearly w.r.t. the network size. Other previous works focus on using attack graphs to illustrate all possible multi-stage and multi-host attack paths, which using the attack graphs as a default attack scenario to develop algorithms for identifying the critical attack assets in a network [21], or establishing scenario attack graphs (SAGs) based on the basic attack graphs to get a whole picture of the current network security situation, and then using such SAGs to predict the possible threats and attacks by correlating detected events or activities [16]. ILLATION does not emulate all possible attack paths in an in-detailed analysis to identify the potential threats and attacks as the way of attack graph analysis model. Instead, ILLATION uses the AI technique to learn from adversaries' behavior and tunes the initial assessment with network constraints that restrict interactions between vulnerabilities and network elements.

3 System and Models

In this section, we introduce ILLATION's overview architecture in Figure 1 and explain how it addresses the design challenges discussed in Section 1.

3.1 Overview

ILLATION provides a network-specified vulnerability prioritization solution, by learning and inferring patterns from network-specified data (including network security data, network profiles, and network vulnerabilities shown in Figure 1). ILLATION assesses vulnerability risk from the criticality of vulnerability exploits and the likelihood of vulnerability exploits. The system design of ILLATION contains four parts: (1) adversary-exploit data processor, (2) risk assessment, (3) network-vulnerability parser, and (4) prioritization reasoning. ILLATION analyzes five types of input data as shown in Figure 1. The CVE data are vulnerability information data, which come from existing vulnerability datasets (e.g., the national vulnerability dataset (NVD) [42]). The exploit data are the reported records of vulnerability exploits, which come from the public dataset (e.g., from ExploitDB [43]) or some organizations' private datasets. The other three types of input data are from the network environment, including the known and suspicious adversary data on a network (network adversaries), the network profile data, and the vulnerability list that needs to be prioritized (network vulnerabilities), which can be obtained by the cyber admins on a network.

First, we illustrate Stage (a) vulnerability risk assessment, where the patterns of *advisories' motivations and abilities* on exploiting vulnerabilities are extracted by the adversary-exploit data processor ((1)). Such patterns are learned by the risk assessment model ((2)). When exploiting vulnerability for attacks, adversaries prefer to maximize their gains by leveraging costs and gains [44]. The *adversary's motivation* for exploiting a vulnerability is identified by CVSS vectors, where the access complexity indicates how difficult to exploit this vulnerability (the costs), and the confidentiality, integrity, and availability impacts indicate how significant the consequences could be (the gains). Thus, such CVSS vectors represent the combinations of costs and gains that motivate adversaries to exploit vulnerabilities. For new vulnerabilities with the same CVSS vector values as the exploited vulnerabilities, the associated criticality of vulnerability exploits might increase. The *adversary's ability* is identified by previous experiences in utilizing vulnerabilities to attack services. When attacking the same service, such previous experiences might reduce the learning costs of using new vulnerabilities. Thus, for new vulnerabilities rooted in the same vulnerable services that have been exploited by adversaries, the associated likelihood of vulnerability exploits might increase.

In the implementation of Stage (a), the AI technique of neuro-symbolic computing [45] is employed for the risk assessment model. The design of (1) and (2) follows our previous work LICALITY on vulnerability risk assessment [13], where a new vulnerability risk score can be assessed by learning risk patterns of advisory's motivation and ability on exploiting vulnerabilities. The neural network side of this risk assessment model learns the patterns of advisories' abilities (for evaluating the likelihood of vulnerability exploits), and the symbolic side (logical programming) learns the patterns of advisories' motivations (for evaluating the criticality of vulnerability exploits).

Then, we present Stage (b) vulnerability prioritization,

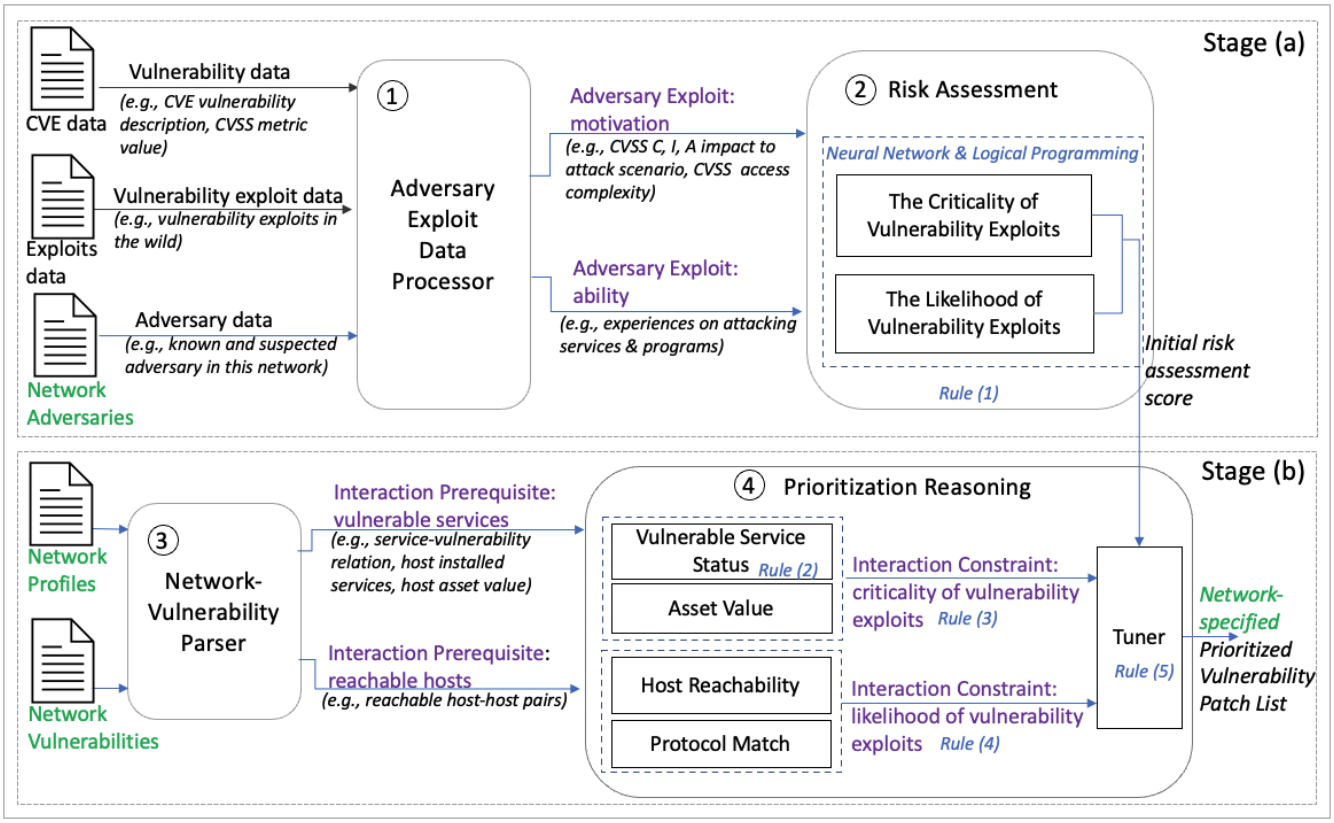


Fig. 1. An overview of ILLATION.

where two *interaction constraints for the criticality of vulnerability exploits and the likelihood of vulnerability exploits* are extracted by the network-vulnerability parser(③) from network profiles and network scanned vulnerabilities. Currently, ILLATION considers *vulnerable services* and *reachable hosts* as two interaction prerequisites for the further prioritization reasoning function. Two *interaction constraints for the criticality and the likelihood of vulnerability exploits* are inferred from these interaction prerequisites facts. To infer the constraint for the criticality of vulnerability exploits, we argue that vulnerabilities are rooted in services or software programs. If no vulnerable service is running on a host machine, the success of a vulnerability exploit is prior impossible [24]. Usually, the larger the asset value of a host, the higher the critical level of its associated vulnerabilities w.r.t. the network. To infer the constraint for the likelihood of vulnerability exploits, we notice that reachability is one of the fundamental prerequisites for modeling network attacks, where both a client-side attack and a server-side attack require an available network patch for attackers to reach the victim's server [20]. Details of ③ are illustrated in Section 3.3.1. Finally, ILLATION's prioritization reasoning model(④) generates a network-specified prioritized vulnerability patch list by tuning the initial risk assessment score with the aforementioned constraints. Details of ④ are illustrated in Section 3.3.3.

In the implementation of Stage (b), ILLATION employs the AI technique of probabilistic logical programming [46] to develop the prioritization reasoning model. In the following sections, the logical representations used in ILLATION contain 1) a set of logical ground probabilistic facts F (in the form of $p :: f$), where p is a probability and f is a ground atom; and 2) a set of logical rules R, which is in the form $h : -b_1, ..., b_n$. The h is an atom and the b_i are literals. The symbol of $(: -)$ represents the logical implication and the symbol of $(,)$ represents the logic conjunction.

3.2 Stage (a): Vulnerability Risk Assessment

In Stage (a), ILLATION initially assesses vulnerability risk by following our previous work LICALITY [13] and specifically considers the known and suspected advisories in the given network.

In ①, first, ILLATION defines adversary attributes that represent the patterns of vulnerability exploits in the given network from both exploits data and network adversaries data. These adversary attributes contain: 1) a set of software services affected by vulnerability exploits, 2) a set of associated CVSS vectors (i.e., access complexity, confidentiality, integrity, and availability impacts), and 3) a set of associated CVE-ID. Then, ILLATION labels a vulnerability in the CVE data as "*more likely to be exploited*" if it matches either one of the following conditions: 1) its vulnerability description contains at least one software service listed in adversary attributes sets; 2) it has the same CVSS vector value listed in adversary attributes sets; 3) its CVE-ID is listed in adversary attributes sets. Otherwise, ILLATION labels a vulnerability as "*less likely to be exploited*". ILLATION also transforms vulnerability descriptions into vector representations by using an NLP tool of latent semantic analysis (LSA) [47]. LSA computes the contextual-usage meaning of words to a

large text corpus and supports discovering hidden features in the data. Finally, ILLATION constructs a dataset with these labels and vector representations, where the patterns of advisories' motivations and abilities on exploits are embedded.

In ②, ILLATION assesses vulnerability risk based on Rule (1), where $likeli\_exploit$ represents the assessed likelihood of exploitation, Lsa represents the LSA feature, $Status$ represents the exploit status of a vulnerability identified by the label in the data set. The $critica\_exploit$ represents the assessed criticality of exploitation, where Ac, C, I, A refers to the CVSS vector of access complexity, confidentiality, integrity, and availability impacts.

$$\begin{aligned} &initial\_assess(Cve\_id, Status) :- \\ &critica\_exploit(Ac, C, I, A, Status), \\ &likeli\_exploit(Cve\_id, Lsa, Status). \end{aligned} \tag{1}$$

For example, for the vulnerability with the CVE-ID of CVE-2019-0611, its initial risk assessment score is represented by $initial\_assess$(2019-0611,1), which represents its initial risk assessment score. The $likeli\_exploit$(cve-2019-0611,lsa,1) represents how likely it would be exploited, and the $critica\_exploit$(h,c,c,c,1) represents how critical if a vulnerability with the same CVSS vector (access complexity is "high", the confidentiality, integrity, and availability impacts are "complete") is exploited.

In the implementation, a risk assessment model developed in the neuro-symbolic computing technique of DeepProbLog [45] learns patterns embedded in the dataset. This risk assessment model is trained by following the parameter learning [45] as 1) the parameters of the neural network side are updated by back-propagation [48] with the gradients of the loss w.r.t. the output of $likeli\_exploit$; and 2) the parameters of the symbolic side are updated by gradient semiring [45] with the gradients of the loss w.r.t. the learnable probabilities in the $critica\_exploit$.

3.3 Stage (b): Vulnerability Prioritization

3.3.1 Interaction Prerequisites

The ③ network-vulnerability parser parses network profiles and network vulnerabilities as a knowledge base with several logic facts, which represent interaction prerequisites. As we discussed in Section 3.1, ILLATION focuses on two fundamental prerequisites (vulnerable services and reachable hosts) of interactions between vulnerabilities and network elements:

- vulnerable services: the vulnerable services on a network, including the logic facts of $vul\_serv\_relation(Cve\_id, Serv)$, $P_s :: host\_serv(Node, Serv, Protocol)$, and $P_v :: asset\_value(Node)$.
- reachable hosts: the host-host reachable pairs on a network, including the logic fact of $P_h :: host\_pairs(Node_1, Node_2, Protocol)$

For vulnerable services, ILLATION extracts semantic information from both network profiles and network vulnerability data. Such semantic information is in the form of logic facts as $vul\_serv\_relation(Cve\_id, Serv)$, $P_s :: host\_serv(Node, Serv, Protocol)$, and $P_v :: asset\_value(Node)$. The $vul\_serv\_relation$ indicates which vulnerabilities are associated with a service. ILLATION parses $vul\_serv\_relation$ from the vulnerability description in network vulnerability data. The $host\_serv$ indicates the installed services' running status on a host, which is extracted from network profile data. The $asset\_value$ represents the user-defined critical level of a host machine in a network, which is assigned by a cyber admin. Usually, assigning a large asset value to a host machine means that cyber admins must ensure the highly secure operation level on this host machine. In other words, exploiting vulnerabilities on that machine is associated with high criticality. For reachable hosts, ILLATION extracts semantic information from network profiles as reachable host-host pairs as $P_h :: host\_pairs(Node_1, Node_2, Protocol)$. The reachable host-host pairs indicate the service level (via the protocol) connections among host machines.

ILLATION defines both P_h and P_s as status indicators with the value of 0 or 1, where $P_h = 1$ means the host-host pair $host\_pairs$ is reachable, and $P_s = 1$ means the vulnerable service $host\_serv$ is running. If the status changes (e.g., changes from reachable to unreachable or from running to not running), the value of these two indicators is 0. ILLATION extracts asset value from network profile data as $P_v :: asset\_value(Node)$, where P_v is a value from 0 to 1. To calculate P_v, the asset value of the host $Node$ is divided by the maximum asset value. For example, in a network, asset values range from 1 to 10, and a host is assigned an asset value of 7. Thus, its P_v is 0.7. But solely relying on asset value to prioritize vulnerability is not appropriate. The asset value can indicate the most vital assets to an organization, but cannot indicate the risks that affect the asset. Remediating the high-risk vulnerabilities that actually affect the most critical assets than patching all vulnerabilities associated with these high-value assets is a more efficient strategy.

3.3.2 Interaction Constraints

To infer the constraint of $constr\_likeli$ and $constr\_critica$, ILLATION defines the $vulnb\_serv\_status$, $host\_pairs$, and $protocol\_match$ as follows. In this section, we illustrate the details of these three logic representations.

- vulnerable service status: a set of services installed on a host machine in the network as $vulnb\_serv\_status(Cve\_id, Node, Protocol_s)$.
- host reachability (overall): the overall reachability status of a host on a network as $host\_pairs(\_, Node, Protocolr)$.
- protocol match: the approximate percentage of how host reachability affects vulnerability exploits $P_m :: protocol\_match(Protocol_r, Protocol_s, Node)$.

First, ILLATION defines $vulnb\_serv\_status$ to indicate the vulnerable service status. By following Rule (2), ILLATION identifies the demographic information of a vulnerability on a network, including CVE-ID, associated service, host machine, and communication protocol.

$$vulnb\_serv\_status(Cve\_id, Serv, Node, Protocol_s) :- \\ host\_serv(Node, Serv, Protocol_s), \\ vul\_serv\_relation(Cve\_id, Serv). \quad (2)$$

In Rule (2), $host\_serv$ represents a service's status (running or not running) in a machine, and $vul\_serv\_relation$ represents which vulnerability is associated with which service. For example, in Appendix A, the test case example shows that Microsoft Edge is running on the host of user2. ILLATION parses such network profile data as the logic fact $1 :: host\_serv(user2, ms\_edge, https)$. The logic representation $vul\_serv\_relation(cve-2019-0611, ms\_edge)$ indicates that a vulnerability CVE-2019-0611 can be exploited in a running Microsoft Edge service. By following Rule (2), ILLATION obtains the value of $vulnb\_serv\_status(cve-2019-0611, ms\_edge, user2, https)$, which means that vulnerability CVE-2019-0611 on the host of user2 is vulnerable, and could be exploited in the current network. Then, ILLATION uses a logic representation $host\_pairs(\_, Node, Protocol)$ to indicate the reachability status of a vulnerable host $Node$ via $Protocol$. For example, if $host\_pairs(\_, user2, https)$ is true, it means that other hosts can reach the user2 via the protocol HTTPS.

Similarly, the previous work of attack graph [14] also uses host-host pair at the protocol level as a host access control list (HACL) to represent all allowed accesses between hosts on a network. The attack graph infers multi-hop network access of attackers from the given network configurations on this HACL and attackers' privilege after exploiting vulnerabilities on each host. The attack graph is an **in-detailed** analysis of *how* a vulnerability exploit interacts with attackers' network access, which is a tool to draw a concrete map of attack paths. *Being different from the attack graph, ILLATION uses these host-host pairs to identify which host has restrictions on being reached by other hosts* ***without computing interactions in detail.*** We notice that one of the most popular mitigation strategies is to reduce the probability of connections between neighbor hosts and the suspicious host. For example, the mitigation actions (e.g., creating filtering rules, changing IP address, and blocking the port) restrict the possible layer-3 attacks (e.g., the distributed denial-of-service (DDoS) attack [49]) from neighbor hosts to the suspicious host by reducing the possibility of host-host connections [16]. Based on this finding, ILLATION focuses on capturing the specious host's reachability status w.r.t. all other hosts globally. Host-host reachable pairs can be inferred from network configurations (e.g., firewall rules, access control policies, security groups, network topology, etc.). In the implementation, there are many network reachability analyzers (e.g., AWS network reachability analyzer [50]) available for users, which can directly generate the results of host-host reachable pairs on a network.

Then, ILLATION interprets how the host reachability affects vulnerability exploits from different connection levels of high, medium, and low. In the logic representation $P_m :: protocol\_match(Protocol_r, Protocol_s, Node)$, $Protocol_r$ indicates the protocol required by a host-level connection between the target host and other hosts, and $Protocol_s$ indicates the protocol required by a vulnerable service-level connection between a vulnerable service running on the target host and other hosts. For example, $vulnb\_serv\_status(cve-2019-0611, ms\_edge, user2, https)$ means that a vulnerable service Microsoft edge on the host of user2 can connect to other hosts via P_s HTTPS, and $host\_pairs(\_, user2, https)$ means that the host of user2 can connect to other hosts via P_r HTTPS.

The value of P_m indicates three categorical risk degrees associated with these three connection levels respectively. ILLATION defines the value of P_m based on the value of the target distribution metric in the CVSS environmental metrics [51], which indicates the different proportions of vulnerable systems. If $Protocol_r$ matches $Protocol_s$, $P_m = 1$ (high); if these two protocols do not match, and the host $Node$ can be reached by other hosts (e.g., as $1 :: host\_pairs(\_, webServer, https)$), $P_m = 0.75$ (medium); and otherwise, $P_m = 0$ (none). $P_m = 1$ means the direct connection from both this vulnerable service and target hosts to other hosts. $P_m = 0.75$ means the direct connection from the target host to other hosts. In this case, if adversaries compromise this vulnerable service, due to the lack of direct connection from this vulnerable service to other hosts, adversaries might need extra efforts to launch further attacks from this vulnerable service to other hosts. $P_m = 0$ means there is no connection that exists based on host reachability status, where the target node is isolated on a network. Additionally, ILLATION considers the relationships between protocols on different layers of the network stack. For example, in the TCP/IP protocol stack layers [52], TCP can support application-layer protocols SMTP, FTP, and HTTP. ILLATION defines a subset condition as $TCP \supset \{SMTP, FTP, HTTP\}$ to represent such layer-based dependency. Therefore, if $Protocol_r = TCP$ and $Protocol_s = HTTP$, $P_m = 1$.

Finally, ILLATION infers the constraint of $constr\_critica$ in Rule (3) and $constr\_likeli$ in Rule (4) based on these logic representations discussed above. In Rule (3), the $constr\_critica$ represents how critical the user-defined consequence of vulnerability exploits on a host is. In Rule (4), the $constr\_likeli$ represents how likely a running service on a host can be reached by other hosts in a network.

$$constr\_critica(Cve\_id, Serv, Node, Protocol_s) :- \\ asset\_value(Node), \\ vulnb\_serv\_status(Cve\_id, Serv, Node, Protocol_s). \quad (3)$$

$$constr\_likeli(Node, Protocol_r, Protocol_s) :- \\ host\_pairs(\_, Node, Protocol_r), \\ protocol\_match(Protocol_r, Protocol_s, Node). \quad (4)$$

3.3.3 Prioritization Reasoning

ILLATION tunes the initial risk assessment with the interaction constraints $constr\_likeli$ and $constr\_critica$. These two constraints range from 0 to 1, which might reduce the initial risk assessment. The ④ *prioritization reasoning*, firstly, infers the interaction constraints $constr\_likeli$ and $constr\_critica$ from the knowledge base of interaction prerequisites (details of these prerequisites and constraints are discussed in Section 3.3.1 and Section 3.3.2). Such constraints

indicate which vulnerable service is running, how critical the service is assigned by cyber admins, which vulnerabilities are associated with this service, and to which degree a vulnerable service can affect other hosts on a network. Then, ④ generates a network-specified prioritization vulnerability path list by tuning the initial risk assessments with these two constraints as Rule (5), which presents the prioritization reasoning model as:

$$\begin{aligned} vul\_risk(Cve\_id, Serv, Node) :- \\ initial\_assess(Cve\_id, 1), \\ constr\_critica(Cve\_id, Serv, Node, Protocol_s), \\ constr\_likeli(Node, Protocol_r, Protocol_s). \end{aligned} \tag{5}$$

where $initial\_assess$ is the initial risk assessment. ④ *prioritization reasoning* generates a network-specified risk assessment score $vul\_risk(Cve\_id, Serv, Node)$ for each given vulnerability w.r.t. its associated running service and host node. A vulnerability that has the highest risk score will first be recommended for remediation.

3.4 Network Defense Assumptions for ILLATION

ILLATION is developed under three network defense assumptions. Assumption 1: the network reachability among hosts can be controlled through network defense rules and access control policies, and results in the reachability status change of the host-host pairs; Assumption 2: ILLATION only considers the direct vulnerability exploitation. For example, this study does not consider multi-hop attach-chain scenarios; Assumption 3: there is no defense rule (i.e., firewall policy/flow rule) conflicts when performing the mitigation. For example, overlapping firewall rules are not allowed in the entire network system. In this study, based on Assumptions 1 & 2, any given vulnerability risk can be adjusted by taking a mitigation action to block access to the vulnerable services. Based on Assumption 3, changing one defense rule for mitigating a vulnerability on a host will not affect the vul_risk of vulnerabilities on other hosts in the network in Rule (5).

4 Evaluation

4.1 ILLATION Implementation

We have implemented ILLATION on a Linux PC with two Intel(R) Xeon(R) CPU @2.30GHz processors and 26GB RAM. ILLATION is developed in Python 3.7, the neuro-symbolic computing technique of DeepProblog [45] and the logical programming language ProbLog [46], with around 2,500 lines of code in total.

The implementation of ILLATION contains the following key steps. In Stage (a), we use our previous work LICALITY [13] to develop a risk assessment model (as shown in Stage (a)). A dataset $\mathcal{D}$ is constructed with the data instance (including CVE-ID, vulnerability descriptions, and CVSS vector) and the embedded risk patterns of the test case (as labels discussed in Section 3.2). This dataset contains 155,176 vulnerabilities from NVD [42], and the exploits records from ExploitDB [43]. The text data of vulnerability descriptions are transformed as vector representation via NLP techniques of LSA [53]. To train the risk assessment model, we randomly sampled the training set, validation set, and testing set from the dataset $\mathcal{D}$ (e.g., at the rate of 80%, 10%, and 10%) by excluding vulnerabilities for labeling and evaluation that are covered in evaluation cases. The output of this risk assessment model is saved as $initial\_assess$, which will be used in Rule (5) for computing the network-specified risk assessment score. In Stage (b), the network-vulnerability parser parses the given input of network profiles and network vulnerabilities as a knowledge base for the interaction prerequisites as $host\_pairs$, $host\_serv$, and $asset\_value$. The logic models in Rule (2), Rule (3), and Rule (4) infer the interaction constraints from these prerequisites logic facts, and the logic model in Rule (5) tunes the initial risk assessment score with constraints to output network-specified risk score.

4.2 Experimental Environment Settings

Due to the limitations of cloud resources, we deploy two network environments: a case study network environment deployed on AWS EC2 [54], and a simulated network environment by using Mininet [55]. Mininet is a network simulator that allows users to create a network with virtual hosts, switches (supporting the OpenFlow protocols), controllers, and links. Mininet also has been widely used in previous work as a security experiment platform [16], [56].

4.2.1 Case Study

In this study, we compare ILLATION with the most popular vulnerability prioritization method CVSS [10] on prioritizing the "new" vulnerabilities in the test case. Both CVSS base metrics and environmental metrics (shown as CVSS B&E score in Table 1) are used to assess vulnerabilities in this test case. The CVSS environmental metrics are designed to assess the potential impact of a vulnerability in the end user's network environment, where countermeasures can be taken to reduce the effect or likelihood of vulnerability exploits. The CVSS environmental metrics contain collateral damage potential (CDP), target distribution (TD), and confidentiality, integrity, and availability requirement (CR/IR/AR). The CDP measures the potential for loss, the TD measures the proportion of vulnerable systems, and the security requirements of the CIA triad can be adjusted to its business requirements as well. We design a test case from real adversary data as:

- Network Adversaries: this data is the advanced persistent threat (APT) adversary APT28 [57] reported in 2017.
- Network Profiles: the host reachability of network profiles before & after mitigation are marked as NP-1 and NP-2 in Figure 2. In NP-1, all 5 hosts are connected via protocols. In NP-2, mitigation actions are taken on the host of User2, where the connection via HTTPS is blocked.
- Network Vulnerabilities: this data is a vulnerability list with 5 vulnerabilities from APT attack in 2020 [58] and 5 unexploited vulnerabilities in the dataset $\mathcal{D}$. We purposely select the APT vulnerability and the unexploited vulnerability with similar CVSS vectors & base scores for each host to illustrate ILLATION's performance in learning adversary risk patterns.

In this test case, we compare ILLATION with CVSS to answer the following questions:

1) How does ILLATION's network-specified evaluation match the assessment patterns of the CVSS environmental metric model in NP-1 and NP-2?
2) When changing the network environment, how does ILLATION respond to such changes and result in changes in vulnerability risk assessment?

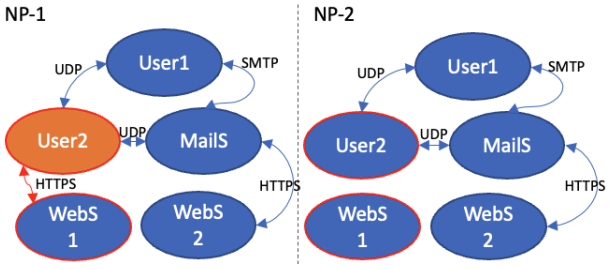


Fig. 2. The Host Reachability in NP-1 and NP-2.

The network environment of the case study is deployed in a 5-node Kubernetes [25] cluster hosted in AWS EC2. ILLATION's scope does not cover scanning and configuring the value of network assets. In this paper, ILLATION utilizes reports generated by the existing cloud tools, such as Qualys [59], AWS network reachability analyzer [50], etc.

4.2.2 The Simulated Networks

In the simulated network experiment, we simulate a network environment in a private cloud environment by using Mininet [55] and randomly distribute 69,730 vulnerabilities that are associated with the top 50 products in CVEDetails [60] to this network. The emulated software-defined network (SDN) [61] uses the Open Network Operating System (ONOS) [62] controller, where mitigation actions (at the layer-2 and layer-3) can be captured. Due to the limitation on the number of virtual hosts (up to 100 with our computing resource) emulated by Mininet, a Python script is developed to simulate the network flow tables for 500, 1k, 5k, and 10k virtual hosts.

In evaluation, vulnerabilities are randomly selected from 69,730 vulnerabilities associated with the top 50 products in CVEDetails [60]. These products cover the common modern operation systems (OS) (i.e., Windows, Linux distributions, and Mac OS), popular applications (e.g., Chrome, IE, Edge, Safari, Firefox, etc.), and server software (e.g., databases, web servers, etc.). Based on operating system compatibility, network service, and opened ports, each host is assigned 0 to 5 operation system vulnerabilities. There are 0 to 15 application vulnerabilities associated with the various applications discussed above.

4.3 Results

4.3.1 The Case Study

In this evaluation, in Stage (a), the risk assessment model is trained to learn the risk patterns of the given network adversary. The AUC (Area Under the Receiver Operating Characteristics (ROC) Curve) of the risk assessment model is 0.93. Details of the model training/validating/testing process are illustrated in our previous work LICALITY [13]. In Stage (b), as shown in Figure 2, NP-1 represents the host reachability status before mitigation, and NP-2 represents such status after mitigation, where the HTTPS connection has been blocked between User2 and WebS1. Both NP-1 and NP-2 are snapshots of the given network environment. The time to take such a snapshot could be after taking a mitigation action for a single vulnerability or after taking multiple mitigation actions for a set of vulnerabilities (e.g., the top 50 vulnerabilities in the network have been ranked), which depends on the user-defined mitigation strategy and cost. In general, fine-grained updates of the network profile imply higher accuracy in a network environment but often come with higher operational costs. Each new snapshot of the network environment is a new network profile. In the implementation, ILLATION updates constraints by running Rule (2), Rule (3), and Rule (4) when giving a new network profile input. In this case study, compared with NP-1, ILLATION parses the network profile of NP-2 by adjusting these logic facts of $host\_pair(user2, webs1, https)$ and $host\_pair(webs1, user2, https)$ to be unreachable, and adding new logic facts of $protocol\_match$ to indicate the current connection status of user2 and webs1. Details are shown in Appendix A. When using CVSS B&E metrics, the categorical value of CDP is based on the host's asset value (1-10), where None (1-2), Low (3-4), Low-Medium (5-6), Medium-High (7-8), and High (9-10). Table 2 shows the details of encoded categorical values for CVSS environmental metrics in the case study evaluation. For example, if a mitigation action is taken on the host of user2 by blocking part of its connections, the TD will be adjusted from high to medium. The CVSS online calculator[2] is used to calculate vulnerability score with the given network profiles NP-1 and NP2.

The first three columns of Table 1 show the vulnerability CVE-ID and the host information, where APT vulnerabilities are in bold. The other columns contain vulnerability risk scores assessed by ILLATION and CVSS Base&Environmental Metrics [51] in NP-1 and NP-2 respectively. The CVSS B&E score of vulnerability is a network-specified CVSS score. By giving NP-2, both CVSS B&E metrics and ILLATION capture changes in network reachability on User2 and WebS1. When using the method of CVSS B&E metrics, the vulnerability's TD and AR values have been adjusted to match the network environment in NP-2. When using ILLATION, the logical representations of host reachability status and protocol match have been updated. Comparing the change in vulnerability prioritization ranks, ILLATION reflects the pattern of changes in vulnerability scores and prioritization ranks as same as the CVSS B&E metric model. Compared with the prioritization list in NP-1, in NP-2, the rank of the highest risk vulnerability CVE-2019-11510 (in NP-1) and the ranks of vulnerabilities on the isolated host node WebS1 have been reduced due to the network constraints on both the likelihood ($host\_pairs$) and the criticality ($vulnb\_serv\_status$) of exploits. Additionally, ILLATION can better learn the network adversary's risk patterns, and prioritize APT vulnerability on the top for vulnerabilities with the same/similar CVSS base score and network configurations. The live demo for the case study

2. CVSS V2 online calculator:https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator

TABLE 1
Comparing Risk Scores and Ranks of Vulnerabilities by Using CVSS and ILLATION.

Host	Asset Value	CVEs	CVSS Base Score [51])	CVSS B&E Score [51]: NP-1 (rank)	ILLATION Score: NP-1 (rank)	CVSS B&E Score [51]: NP-2 (rank)	ILLATION Score: NP-2 (rank)
User1	6	CVE-2020-7995	10	8.7 (7)	0.5995 (7)	8.7 (3) (↑)	0.5995 (6) (↑)
User1	6	**CVE-2020-5902***	10	8.7 (7)	**0.5998 (6)**	8.7 (3) (↑)	**0.5998 (5)** (↑)
User2	10	CVE-2019-0611	7.6	8.8 (6)	0.5456 (8)	6.6 (8) (↓)	0.4092 (8) (−)
User2	10	**CVE-2019-11510***	7.5	9.4 (1)	**0.9988 (1)**	7.1 (7) (↓)	**0.7491 (4)** (↓)
MailS	8	CVE-2020-2950	7.5	9.3 (2)	0.7985 (4)	9.3 (1) (↑)	0.7985 (2) (↑)
MailS	8	**CVE-2019-19781***	7.5	9.3 (2)	**0.7993 (3)**	9.3 (1) (↑)	**0.7993 (1)** (↑)
WebS1	9	CVE-2004-0248	6.8	9.1 (4)	0.1396 (10)	0 (9) (↓)	0 (10) (−)
WebS1	9	**CVE-2020-1631***	6.8	9.1 (4)	**0.8969 (2)**	0 (9) (↓)	**0 (10)** (↓)
WebS2	8	CVE-2014-1637	5	7.6 (9)	0.4863 (9)	7.6 (5) (↑)	0.4863 (7) (↑)
WebS2	8	**CVE-2018-13379***	5	7.6 (9)	**0.7985 (4)**	7.6 (5) (↑)	**0.7985 (2)** (↑)

* APT attack vulnerability.
The arrow after each ranking value represents the change of rank (increasing/decreasing) compared to the one in the previous column.

evaluation is available on the online logic program platform.[3]

TABLE 2
Details of Network Profiles Encoded in the CVSS Environmental Metrics.

CVEs	Protocol	Host	CDP/TD/CR/IR/AR (NP-1)	CDP/TD/CR/IR/AR (NP-2)
CVE-2020-7995	SMTP	User1	LM/H/L/L/L	LM/H/L/L/L
CVE-2020-5902	UDP	User1	LM/H/L/L/L	LM/H/L/L/L
CVE-2019-0611	HTTPS	User2	H/H/H/H/H	H/M/H/H/H
CVE-2019-11510	HTTPS	User2	H/H/H/H/H	H/M/H/H/H
CVE-2020-2950	HTTPS	MailS	MH/H/H/H/H	MH/H/H/H/H
CVE-2019-19781	UDP	MailS	MH/H/H/H/H	MH/H/H/H/H
CVE-2004-0248	HTTPS	WebS1	H/H/H/H/H	H/N/H/H/H
CVE-2020-1631	HTTPS	WebS1	H/H/H/H/H	H/N/H/H/H
CVE-2014-1637	HTTPS	WebS2	MH/H/H/H/H	MH/H/H/H/H
CVE-2018-13379	HTTPS	WebS2	MH/H/H/H/H	MH/H/H/H/H

L: low; LM: low-medium; MH: medium-high; H: high

4.3.2 The Simulation Networks

In the simulated networks, the running time of the network-vulnerability parser and the prioritization reasoning engine is recorded to show ILLATION's prioritization reasoning engine's performance in a scalable network environment. This scalable network environment has vulnerable hosts ranging from 10 to 10k and has up to 10k vulnerabilities. The network profile contains flow policies from an SDN controller in the simulated SDN network. A flow rule indicates the flow traffic directions and types that are allowed in this network setup by applying defense mechanisms. The network traffic reachability between hosts is represented as logic facts. For example, the logic fact of $host\_pairs(host1, host2, https)$ indicates that host1 can send packets to host2 through HTTPS. The average node connectivity w.r.t. a host is around 31 in the evaluation. Table 3 shows the running time (in seconds) of the network profiling parser. This parser processes the network setups (up to 10k vulnerable hosts) listed in Table 3 within 5 seconds. We also test the running time of the network profiling parser for a large-size network setup data (12.65GB), its running time is around 622 seconds. By tuning the initial risk assessment scores with network constraints, the prioritization reasoning engine assesses vulnerability risk as Rule (5).

3. https://dtai.cs.kuleuven.be/problog/editor.html#task=prob&hash=8b7de28aaa1856109f24ab57adcdf5b1

TABLE 3
Network Profiling Parser Running Time (in Second) in a Scalable Network System.

Vulnerable Hosts in the network	Numbers of Node Connectivity	Network Setups Data Size (in MB)	Network Profiling Parser Running Time (in Second)
10	48	0.009	0.609
50	447	0.079	0.61
100	1261	0.24	0.911
500	8108	1.6	1.01
1K	23405	4.6	1.11
5K	124447	25.2	2.21
10K	364476	74.1	4.13

TABLE 4
Prioritization Reasoning Engine Running Time (in Second) on Assessing Vulnerabilities in a Scalable Network System.

Vulnerable Hosts	10 Vuls	50 Vuls	100 Vuls	500 Vuls	1K Vuls	5K Vuls	10K Vuls
10	35.9	42.9	50.2	106	181	664	1248
50	36.2	44.4	51.5	107	184	669	1270
100	36.4	43.5	53.6	110	186	674	1255
500	38.6	45	54.2	122	195	693	1233
1K	40.2	46.4	56.8	123	218	725	1249
5K	55	61	73	142	231	931	1458
10K	93	100	109	175	271	944	1909

Based on the observation of running time in Table 4, the reasoning engine's running time is affected by the number of vulnerable hosts and the number of vulnerabilities. The prioritization reasoning engine's running time increases when either the number of vulnerable hosts increases or the number of vulnerabilities increases. ILLATION's reasoning engine can assess 500 vulnerabilities in the network with thousands of vulnerable hosts within 3 minutes (up to 175 seconds). When increasing the vulnerabilities assessed to 1k, our reasoning engine can handle it in around 4.5 minutes (up to 271 seconds).

To investigate how the number of vulnerable hosts and vulnerabilities affect the reasoning engine's running time, the average running time for assessing a vulnerability is shown in Figure 3. Due to space limits, we mark the average running time for four testing scenarios (with 10, 50, 100, and 10000 vulnerabilities) in this chart. Figure 3

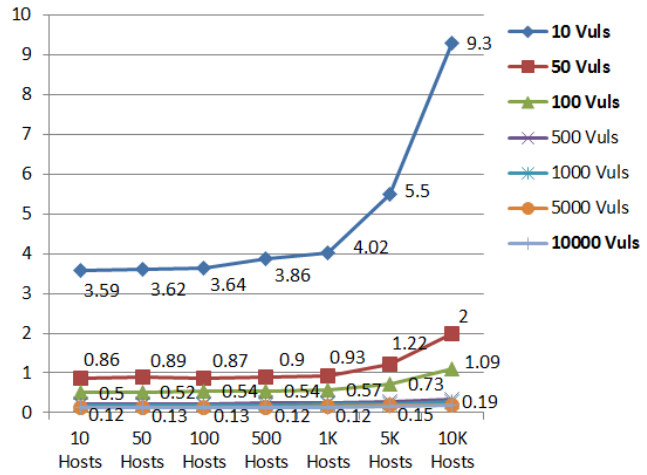


Fig. 3. The Average Running Time (in Second Per Vulnerability) of Assessing a Vulnerability in the Evaluation

shows that in a network with 10-10k hosts, the curve of average running time per vulnerability becomes flattened when assessing more vulnerabilities. For example, when assessing 10k vulnerabilities, the average running time per vulnerability changes from 0.12 seconds (in 10 hosts) to 0.19 seconds (in 10k hosts), which includes the time to parse the network profiles and obtain the constraints. While, for 10 vulnerabilities, it changes from 3.59 seconds (in 10 hosts) to 9.3 seconds (in 10k hosts). A significant increment of the average running time per vulnerability is observed after increasing the number of hosts over 1k. This observation matches the expected behavior of our reasoning engine, because the node connectivity data size mainly affects the running time, in this evaluation, the networks with less or equal to 500 hosts have less than 8.2k node connectivity, while the networks with 1k, 5k, and 10k hosts have 23k, 124k, and 364k node connectivity, respectively, as shown in Table 3. Another factor that mainly affects the total running time is the number of vulnerabilities. This reasoning engine is based on the Sentential Decision Diagram (SDD) [63], which is a tree structure that defines the logical and probabilistic relationships among nodes for reasoning. Thus, to assess a vulnerability, ILLATION must construct an SDD tree structure to conduct reasoning. Assessing more vulnerabilities is usually associated with constructing more SDD tree structures, which then results in a longer running time. Figure 3 also shows that the average running time per vulnerability for assessing 10k vulnerabilities is almost at the lowest level among all network setups. In the scenario of assessing 10k vulnerabilities in the network with 10k hosts, the average running time is 0.19 seconds per vulnerability.

5 Discussion

5.1 Network Data Source

As we explained in Section 4.1, ILLATION's scope does not cover configuring the network and scanning both vulnerability and host reachability. In this section, we introduce some existing tools that can be used to obtain network data for two prerequisites of vulnerable service and host reachability.

The visibility of cloud inventory is critical to ensure consistency in the security and compliance policies. Existing solutions are able to scan the user-defined cloud network and sync inventory frequently in the cloud platforms (e.g., AWS, GCP, Azure, etc.). For example, Qualys [64] can continuously discover cloud inventory in AWS. The user's cloud network inventory (e.g., AWS instances and metadata) can be synced from all AWS regions. All user network assets are tracked by the instance IDs. Therefore, even if users change the asset name, all assigned assets are well-tracked. Additionally, existing solutions allow users to modify asset values, where users can enter values for assets based on the business impact [59]. Similarly, Google cloud asset inventory tool [65] can discover, monitor, assign business value, and analyze all users' cloud assets as well. ILLATION utilizes these inventory data as the data source for network profiles, which in turn help to identify hosts, the services installed on them, and their associated asset value in the logic reasoning model.

Network reachability data indicates the connectivity path between a source and a destination resource in the VPCs. ILLATION utilizes the host-host reachable pair data to generate overall network reachability in the logic reasoning model. In the public cloud platform (i.e., AWS, GCP), the reachability analyzer can identify the reachable/unreachable paths of the destination port between two virtual machines by analyzing multiple cloud environment configurations in a VPC, such as the security group, network ACL, route table, and load balancer [50].

5.2 The Extension Capability of ILLATION

The good extension capability of the logic model has been highlighted in the previous study [66]. Although in this paper, ILLATION only parses the host reachability from network profile data, the model of ILLATION still has the potential to cover more in-detailed reasoning on a host-level connection. Following the assumption of attack graph [14], where a successful vulnerability exploit can be indicated by the escalation of privilege, ILLATION can add logic representation of access control to explicitly represent the access control policy. By defining the access control policy with accounts (e.g., with the client, user, root privilege, etc.), ILLATION can be extended to capture the change of privilege on a host machine. Additionally, ILLATION can be extended to reflect the layer-2 mitigation actions (e.g., isolating a suspicious host by changing its MAC address [16]) to mitigate vulnerabilities regarding layer-2 attacks (e.g., the ARP Spoofing attack). By building upon the basic model, ILLATION has the capacity to analyze more complex scenarios with more input features in future studies.

6 Conclusion

This paper presents a network-specified vulnerability prioritization method to enhance vulnerability risk prioritization by successfully incorporating adversary behavior and interaction constraints into vulnerability assessment. ILLATION is a neural network and logical programming-based integrated solution for vulnerability risk prioritization. ILLATION explicitly learns adversaries' motivation and ability

on a network, and tunes the initial risk assessment score with the constraints of interactions between vulnerability and network elements. Evaluation results show that ILLATION effectively learns from the network. The overall result is promising and shows the potential to handle vulnerability prioritization tasks with a large number of vulnerabilities.

REFERENCES

[1] Verizon Corp. Data breach investigation report (2021), 2021. Last accessed 15 Oct 2021.

[2] Kenneth Alperin, Allan Wollaber, Dennis Ross, Pierre Trepagnier, and Leslie Leonard. Risk prioritization by leveraging latent vulnerability features in a contested environment. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 49–57, 2019.

[3] CVE Details. Cve details, 2021. Last accessed 29 April 2021.

[4] Emma Woollacott. Measuring risk: Organizations urged to choose defense-in-depth over CVE whack-a-mole, 2021. Last accessed 30 May 2021.

[5] Gartner. The future of vulnerability management is risk-based, 2021. Last accessed 21 May 2021.

[6] Luca Allodi and Fabio Massacci. Comparing vulnerability severity and exploits using case-control studies. *ACM Transactions on Information and System Security (TISSEC)*, 17(1):1–20, 2014.

[7] Peter Mell, Karen Scarfone, and Sasha Romanosky. Common vulnerability scoring system. *IEEE Security & Privacy*, 4(6):85–89, 2006.

[8] Sasha Romanosky Peter Mell, Karen Scarfone. A complete guide to the common vulnerability scoring system version 2.0, 2007. Last accessed 1 March 2021.

[9] The Forum of Incident Response and Security Teams (FIRST). Common vulnerability scoring system v3.0: User guide, 2012. Last accessed 9 December 2020.

[10] Peter Mell, Karen Scarfone, and Sasha Romanosky. A complete guide to the common vulnerability scoring system version 2.0. In *Published by the FIRST-forum of incident response and security teams*, volume 1, page 23, 2007.

[11] The Forum of Incident Response and Security Teams (FIRST). Common vulnerability scoring system sig, 2021. Last accessed 21 April 2021.

[12] National Institute of Standards and Technology (NIST). Vulnerability metrics, 2023. Last accessed March 26, 2023.

[13] Zhen Zeng, Zhun Yang, Dijiang Huang, and Chun-Jen Chung. Licality—likelihood and criticality: Vulnerability risk prioritization through logical reasoning and deep learning. *IEEE Transactions on Network and Service Management*, 19(2):1746–1760, 2021.

[14] Xinming Ou, Sudhakar Govindavajhala, Andrew W Appel, et al. Mulval: A logic-based network security analyzer. In *USENIX security symposium*, volume 8, pages 113–128. Baltimore, MD, 2005.

[15] Yeu-Pong Lai and Po-Lun Hsia. Using the vulnerability information of computer systems to improve the network security. *Computer Communications*, 30(9):2032–2047, 2007.

[16] Chun-Jen Chung, Pankaj Khatkar, Tianyi Xing, Jeongkeun Lee, and Dijiang Huang. Nice: Network intrusion detection and countermeasure selection in virtual network systems. *IEEE transactions on dependable and secure computing*, 10(4):198–211, 2013.

[17] Melanie Tupper and A Nur Zincir-Heywood. Vea-bility security metric: A network security analysis tool. In *2008 Third International Conference on Availability, Reliability and Security*, pages 950–957. IEEE, 2008.

[18] Miles A McQueen, Wayne F Boyer, Mark A Flynn, and George A Beitel. Time-to-compromise model for cyber risk reduction estimation. In *Quality of protection*, pages 49–64. Springer, 2006.

[19] Sushil Jajodia, Steven Noel, and Brian O'berry. Topological analysis of network attack vulnerability. In *Managing cyber threats*, pages 247–266. Springer, 2005.

[20] Kyle Ingols, Matthew Chu, Richard Lippmann, Seth Webster, and Stephen Boyer. Modeling modern network attacks and countermeasures using attack graphs. In *2009 Annual Computer Security Applications Conference*, pages 117–126. IEEE, 2009.

[21] Reginald E Sawilla and Xinming Ou. Identifying critical attack assets in dependency attack graphs. In *Computer Security-ESORICS 2008: 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings 13*, pages 18–34. Springer, 2008.

[22] Steven Noel and Sushil Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 109–118, 2004.

[23] Jianping Zeng, Shuang Wu, Yanyu Chen, Rui Zeng, and Chengrong Wu. Survey of attack graph analysis methods from the perspective of data and knowledge processing. *Security and Communication Networks*, 2019:1–16, 2019.

[24] Christopher Kruegel and William Robertson. Alert verification determining the success of intrusion attempts. In *Detection of intrusions and malware & vulnerability assessment, GI SIG SIDAR workshop, DIMVA 2004*. Gesellschaft für Informatik eV, 2004.

[25] Kubernetes.io. Kubernetes Webpage, 2022. Last accessed 5 Oct 2022.

[26] Kelley Dempsey, Eduardo Takamura, Paul Eavy, and George Moore. Automation support for security control assessments: Software vulnerability management. Technical report, National Institute of Standards and Technology, 2020.

[27] Jeff Aboud. Prioritizing vulnerabilities: Why it's essential to an effective vulnerability management program, 2019. Last accessed 21 April 2021.

[28] Rapid7 Corp. Vulnerability Remediation vs. Mitigation: What's the Difference?, 2020. Last accessed 15 Oct 2021.

[29] J Spring, AHE Hatleback, A Manion, and D Shic. Towards improving cvss. *SEI, CMU, Tech. Rep*, 2018.

[30] Ankit Shah, Katheryn A Farris, Rajesh Ganesan, and Sushil Jajodia. Vulnerability selection for remediation: An empirical analysis. *The Journal of Defense Modeling and Simulation*, page 1548512919874129, 2019.

[31] Nikolai Mansourov. Ranking weakness findings, 2018. Last accessed 21 May 2021.

[32] Jack Wallen. Cvss struggles to remain viable in the era of cloud native computing, 2020. Last accessed 21 May 2021.

[33] Christian Fruhwirth and Tomi Mannisto. Improving cvss-based vulnerability prioritization and response with context information. In *2009 3rd International symposium on empirical software engineering and measurement*, pages 535–544. IEEE, 2009.

[34] Jay Jacobs, Sasha Romanosky, Idris Adjerid, and Wade Baker. Improving vulnerability remediation through better exploit prediction. *Journal of Cybersecurity*, 6(1), 2020.

[35] Haipeng Chen, Rui Liu, Noseong Park, and VS Subrahmanian. Using twitter to predict when vulnerabilities will be exploited. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3143–3152, 2019.

[36] Mehran Bozorgi, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 105–114, 2010.

[37] Carl Sabottke, Octavian Suciu, and Tudor Dumitraș. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *24th USENIX Security Symposium*, pages 1041–1056, 2015.

[38] Soumyadeep Hore, Fariha Moomtaheen, Ankit Shah, and Xinming Ou. Towards optimal triage and mitigation of context-sensitive cyber vulnerabilities. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[39] Benjamin L Bullough, Anna K Yanchenko, Christopher L Smith, and Joseph R Zipkin. Predicting exploitation of disclosed software vulnerabilities using open-source data. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, pages 45–53, 2017.

[40] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 273–284. IEEE, 2002.

[41] Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 121–130. IEEE, 2006.

[42] The National Institute of Standards and Technology. NVD: National Vulnerability Database, 2020. Last accessed 25 August 2020.

[43] The Offensive Security. Exploitdb, 2020. Last accessed 25 August 2020.

[44] Luca Allodi, Fabio Massacci, and Julian M Williams. The work-averse cyber attacker model: Theory and evidence from two million attack signatures. *Available at SSRN 2862299*, 2017.

[45] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, pages 3749–3759, 2018.

[46] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467. Hyderabad, 2007.

[47] Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

[48] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[49] Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE communications surveys & tutorials*, 15(4):2046–2069, 2013.

[50] Amazon Web Services. What is reachability analyzer?, 2023. Last accessed 28 Feb 2023.

[51] NIST. Cvss version 2 calculator, 2023. Last accessed March 26, 2023.

[52] Oracle Corp. Tcp/ip protocol stack. Last accessed 12 Nov 2021.

[53] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[54] AWS. Amazon ec2: Secure and resizable compute capacity for virtually any workload, 2023. Last accessed March 26, 2023.

[55] Mininet. Mininet:an instant virtual network on your laptop (or other pc), 2021. Last accessed 15 Oct 2021.

[56] Mahmoud Al-Ayyoub, Yaser Jararweh, Elhadj Benkhelifa, Mladen Vouk, Andy Rindos, et al. Sdsecurity: A software defined security experimental framework. In *2015 IEEE international conference on communication workshop (ICCW)*, pages 1871–1876. IEEE, 2015.

[57] The FireEye. APT28: At the Center Of the Storm, 2017. Last accessed July 1 2021.

[58] The Cybersecurity & Infrastructure Security Agency (CISA). Apt actors chaining vulnerabilities against sltt, critical infrastructure, and elections organizations, 2020. Last accessed 1 July 2021.

[59] Qualys, Inc. Securing amazon web services with qualys, 2022. Last accessed 25 Feb 2023.

[60] CVEDetails. Top 50 products by total number of "distinct" vulnerabilities, 2021. Last accessed 15 Oct 2021.

[61] Dijiang Huang, Ankur Chowdhary, and Sandeep Pisharody. *Software-Defined networking and security: from theory to practice*. CRC Press, 2018.

[62] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, et al. Onos: towards an open, distributed sdn os. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 1–6, 2014.

[63] Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[64] Qualys. Securing amazon web services with qualys, 2022. Last accessed January 30, 2023.

[65] Google, Inc. Cloud asset inventory, 2022. Last accessed 25 Feb 2023.

[66] Eugen Bacic, Michael Froh, and Glen Henderson. Mulval extensions for dynamic asset protection. Technical report, CINNABAR NETWORKS INC OTTAWA (ONTARIO), 2006.

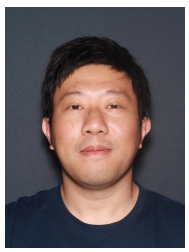
Zhen Zeng (Member, IEEE) received the Ph.D. degree in computer science from Arizona State University in 2022. She is an Assistant Professor of Computer Science at the College of Engineering & Applied Science, University of Wisconsin Milwaukee. Her research interests include vulnerability management, network security, cloud security, AI for cybersecurity, and cybersecurity education. She worked in the high-tech industry for years.

Dijiang Huang (Senior Member, IEEE) received the B.S. degree from Beijing University of Posts and Telecommunications, China, and the M.S. and Ph.D. degrees from the University of Missouri, Kansas in 1995, 2001, and 2004 respectively. He is an Associate Professor with the School of Computing Augmented Intelligence (SCAI), Arizona State University. His research interests include computer networking, security, and privacy. He is an Associate Editor of the Journal of Network and System Management (JNSM) and the IEEE Transaction of Network and System Management. He has also served as the chair at multiple international conferences and workshops. His research was supported by the NSF, ONR, ARO, NATO, and Consortium of Embedded System (CES). He was the recipient of the ONR Young Investigator Program (YIP) Award.

Guoliang Xue (Member 1996, Senior Member 1999, Fellow, 2011) is a Professor of Computer Science at Arizona State University. His research interests span the areas of wireless networking, security and privacy, and optimization. He received the IEEE Communications Society William R. Bennett Prize in 2019. He has served as VP-Conferences of the IEEE Communications Society, and an editor for IEEE Transactions on Mobile Computing and IEEE/ACM Transactions on Networking. He is the Steering Committee Chair of IEEE INFOCOM.

Yuli Deng is a lecturer at GOEE and SCAI in ASU. His research interests include adaptive learning technology in cybersecurity education and the advancement of cybersecurity technologies through the application of multi-disciplinary approaches such as artificial intelligence and machine learning. He received his doctoral degree in Computer Science from Arizona State University in 2021.

Neha Vadnere is a Ph.D. student in Computer Science from Arizona State University, Tempe, AZ, USA. She received M.Tech in Computer Science Engineering from Vellore Institute of Technology (VIT), Vellore, India in 2016 and B.E. in Information Technology from Pune University, India in 2013. She has worked as Network Software Engineer for Intel, Bangalore, India from 2016-2019, and Graduate Intern at Intel from 2015-2016. Her research interests include Software Defined Networking, Policy management, and application of AI and Machine Learning in the field of IoT and Cloud.



Liguang Xie (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Virginia Tech, Blacksburg, VA, USA, in 2013. His research interests are cloud networking, software-defined networking, distributed systems, cloud computing and AI systems.