LICALITY—Likelihood and Criticality: Vulnerability Risk Prioritization Through Logical Reasoning and Deep Learning

Zhen Zeng, *Graduate Student Member, IEEE*, Zhun Yang, Dijiang Huang, *Senior Member, IEEE*, and Chun-Jen Chung

***Abstract*—Security and risk assessment aims to prioritize detected vulnerabilities for remediation in a computer networking system. The widely used expert-based risk prioritization approach, e.g., Common Vulnerability Scoring System (CVSS), cannot realistically associate vulnerabilities to the likelihood of exploitation. The CVSS metrics are calculated from static formulas, and cannot easily integrate attackers' motivations and capabilities w.r.t. the network environmental factors. To address this issue, this paper proposes LICALITY, a vulnerability risk prioritization system. LICALITY captures the attacker's preference on exploiting vulnerabilities through a threat modeling method, and learns threat attributes that contribute to the exploitation of vulnerability. LICALITY creatively uses a neuro-symbolic model, with neural network (NN) and probabilistic logic programming (PLP) techniques, to learn such threat attributes. The risk of vulnerability is assessed from the criticality of exploitation and the likelihood of exploitation. LICALITY consolidates these two measurements by using a logic reasoning engine. In the evaluation, the historical threat and future threat are from real attack scenarios. The results reveal that LICALITY reduces the vulnerability remediation work of the future threat required by the CVSS by a factor of 2.89 in the first case study and by a factor of 1.85 in the second case study. Such future threats are identified as the top routinely exploited vulnerabilities and the APT attack chained vulnerabilities reported in the Cybersecurity and Infrastructure Security Agency (CISA) alerts.**

***Index Terms*—Vulnerability management, risk prioritization, threat model, neural network, logical reasoning, neuro-symbolic.**

I. Introduction

VULNERABILITY management can enhance computer and network systems' security by identifying, evaluating, and mitigating vulnerabilities in a networked system. Risk prioritization is a critical procedure for vulnerability management, where the primary purpose of prioritizing risks is to form a basis for allocating resources [1]. As shown by the recent study, 43% of cybersecurity professionals reported that vulnerability prioritization is one of the most challenging issues in vulnerability management [2]. In practice, the Common Vulnerability Scoring System (CVSS) base score [3], [4] has been widely used. The CVSS score is derived from a static formula based on experts' knowledge, and does not realistically represent the actual exploits [5], [6]. The previous study reveals that "*fixing a vulnerability just because it was assigned a high CVSS score is equivalent to randomly picking ones to fix*" [5]. "*CVSS is designed to measure the severity of a vulnerability and should not be used alone to assess risk* " [7]. The existing machine learning/deep learning-based approaches assess risk by estimating how vulnerabilities' features are associated with the exploits in the wild [5], [6], [8], [9]. However, this approach has limitations that the estimation of risk varies on different datasets and machine learning/deep learning methods [10].

In this paper, we propose a new vulnerability risk prioritization system, called LICALITY, by assessing the riskfrom the "LIkelihood and critiCALITY" of exploitation. LICALITY improves the existing risk prioritization solutions by developing a novel threat modeling method, which is based on the three arguments (A1,A2,A3) discussed in Section IV. This threat modeling method captures the attacker's strength and the exploited vulnerabilities' attributes from a given historical threat in a network system. LICALITY (discussed in Section IV) assesses a vulnerability's risk by consolidating the criticality derived from its CVSS features and the likelihood of exploitation derived from its LSA (latent semantic analysis [11]) features. Additionally, LICALITY learns the threat attributes identified by the threat modeling method with a neuro-symbolic model. This neuro-symbolic model (called the NN-PLP model) merges the neural network (NN) and probabilistic logic programming (PLP). The neuro-symbolic computation is a novel AI approach that integrates neural networks with reasoning methods (e.g., logic and probability) [12]. This computation combines learning from the environment (on the neural network side) and reasoning from what has been learned (on the reasoning side) [13]. Finally, LICALITY consolidates these two measurements to derive the prioritization measurement for each newly discovered threat (discussed as a future threat in this study in Section V). To integrate the historical threat into risk prioritization is valid. As

Manuscript received July 19, 2021; revised September 18, 2021 and November 30, 2021; accepted November 30, 2021. Date of publication December 8, 2021; date of current version June 10, 2022. The associate editor coordinating the review of this article and approving it for publication was N. Zincir-Heywood. *(Corresponding author: Zhen Zeng.)*

Zhen Zeng, Zhun Yang, and Chun-Jen Chung are with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: zzeng22@asu.edu; zyang90@asu.edu; cchung20@asu.edu).

Dijiang Huang is with the School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85287 USA (e-mail: dijiang.huang@asu.edu).

Digital Object Identifier 10.1109/TNSM.2021.3133811

stated by the Cybersecurity and Infrastructure Security Agency (CISA) [14], analyzing the threat's *likely intent, capability, and target* characteristics can better guide the risk analysis. An identified threat (for known attackers) can tune the risk prioritization in the contested environment near-real-time [8].

In summary, this work's main contributions are:

1) assessing a vulnerability's risk from the likelihood and criticality of exploitation, and consolidating these two measurements in the NN-PLP model to derive the prioritization measurement for each newly discovered threat by reasoning;
2) defining threat modeling method to encode the threat attributes (including historical threat and exploits record) as a set of logic rules in the PLP model and labels in the dataset to train a neuro-symbolic-based risk model.

To demonstrate how to use LICALITY, 155,176 vulnerabilities data are extracted from the National Vulnerability Database (NVD) (1999-2021) [15]. These data contain vulnerability descriptions and the related CVSSv2 vectors. In this study, the historical threat and future threat are from real attack scenarios published in the Cybersecurity and Infrastructure Security Agency (CISA) alerts in 2015 and 2020, and the FireEye's Advanced Persistent Threat (APT) attack report in 2017. LICALITY reduces the vulnerability remediation work of the future threat required by the CVSS by a factor of 2.89 in Case 1 and by a factor of 1.85 in Case 2, and reduces such remediation work required by the existing ML/DL-based solution [8] by a factor of 2.19 in Case 2. In Case 1, LICALITY reaches the almost same performance as the existing ML/DL-based solution [8].

The rest of the paper is as follows. In Section II, we describe the related work of assessing vulnerabilities' risk and background on features and tools used in our solution. Section III explains the motivation of LICALITY. Section IV provides descriptions of the proposed system and model of LICALITY. Section V demonstrates how to apply LICALITY for risk prioritization and discusses the comparative studies. Section VI concludes this paper. Appendix lists out all acronyms used in this study.

II. Related Work

A. Risk Prioritization Overview

Risk prioritization is one of the vulnerability management key steps.Cyber defenders prioritize and mitigate vulnerabilities that are more likely to be exploited under limited time and resources for remediation actions [16], [17]. In the practice of vulnerability assessment, firstly, defenders identify vulnerabilities in the system. They prioritize vulnerabilities according to their risk levels, e.g., ranking the highest Common Vulnerability Scoring System (CVSS) score [3] as high risk. Then, they remediate the system by mitigating or removing identified vulnerabilities based on their risk rankings. In general, there are two predominant risk prioritization approaches:the expert-based approach [16], [18], [19] and the machine learning/deep learning-based approach [8], [20].

B. The Expert-Based Risk Prioritization

The Common Vulnerability Scoring System (CVSS) contains three groups, which are Base, Temporal and Environmental, and is developed upon expert-based domain knowledge. The CVSS score is widely used in expert-based risk prioritization. The CVSSv2 was launched in 2007 [21], and the CVSSv3 was released in 2015. The CVSSv2 base score ranges from 0 to 10. It is computed by ordinal assignments of the ease and impact of exploitation as $CVSS_{base} = Impact \times Exploitability$ [5]. The CVSSv2 base score defines the severity level of a vulnerability as Low (0-3.9), Medium (4-6.9), and High (7-10). Although the CVSSv3 and CVSSv3.1 are released recently, most vulnerabilities have been determined by CVSSv2 [22]. In practice, there are 13.5% vulnerabilities (over 15k) in NVD with the highest CVSS base scores range (9-10) [8], and only 10-15% publicly known vulnerabilities have a known exploit, and even fewer are weaponized as part of hacking tool-kits [6]. Thus, vulnerabilities with high CVSS base scores do not correlate well with the exploits [5], [6].

Although the CVSS score does not correlate well with attacks in the wild [8], the previous study still agrees that CVSS provides an excellent estimation of vulnerabilities' criticality [23]–[26]. In practice, organizations and companies make their changes to the CVSSv2 scheme to make it work better in their working scenarios [27]. By analyzing the *Symantec* data on real attacks detected in the wild, the CVSSv2 measurements of *Access Complexity* (AC), *Confidentiality Impact* (C), *Integrity Impact* (I), and *Availability Impact* (A) provide *a useful first indicator for exploits in the wild* [24]. The level of Complete represents the highest potential loss if the vulnerability is exploited. There is a clear cut-off distinction about attacks in the wild between vulnerabilities with Low complexity, High impact, and vulnerabilities with High complexity Low impact [24], and a trade-off in vulnerability's impact and complexity measurements [26]. For example, the Work-Averse cyber attackers prefer to exploit Low complexity High impact vulnerabilities [26]. Researchers also study the effect of different CVSS measurements on vulnerability exploit delay, where CVSS measure is used as a condition to filter data [23].

C. The Machine Learning/Deep Learning-Based Risk Prioritization

Researchers attempt to explore better associations between vulnerabilities' features and exploits [28], [29] by using the machine learning (ML)/deep learning (DL) model. The key risk factors of vulnerabilities contain *the probability of compromise, consequence*, and *time* [8]. The existence of exploits is identifiedas a significantly better risk metric than CVSS scores [5], [6], [8], [9]. Multiple databases (e.g., ExploitDB, *Symantec* WINE, NVD, and Twitter data) are used to build a dataset for training and testing machine learning models in these studies [9], [28]–[30]. The NVD is a U.S. government repository of standards-based vulnerability management data and is maintained by the National Institute of Standards and Technology (NIST) [15]. All vulnerabilities in NVD have the

unique Common Vulnerabilities and Exposures (CVE) identification number, and the CVSS base score and vector. However, compared to a consistent estimation from the expert-based CVSS approach, the results from the ML/DL-based approach vary in different input and output datasets and ML/DL models. The possible reasons are: (1) the ground truth of exploits varies case by case [5], [6], [8]; (2) the extracted vulnerabilities' features vary among studies [8]–[10]. The previous study also shows that the selection of training and testing datasets and highly imbalanced attack/non-attack data critically affects the estimated performance of predictive model [10]; and (3) the limitation of tools for processing vulnerabilities' data, e.g., using the Natural Language Processing (NLP) tools. NLP tools are highly domain-specific tools, and all results are rooted in analyzing vocabularies. Thus, the ML/DL-based model that was trained on some existing vocabularies (by using NLP tools to process such text data [8], [9]) might be hard to work on vocabularies that emerged later with high-quality [31].

Due to these various reasons, assessing a vulnerability might vary greatly among different ML/DL-based risk models under different datasets and training techniques. Thus, solely depending on the supervised ML/DL-based model for risk prioritization still has limitations. To overcome these limitations of the existing ML/DL-based approach, our solution integrates the CVSS measurements (AC, C, I, A) as part of risk metrics through threat modeling. We develop a solution with neuro-symbolic computing (the NN-PLP model) to learn the threat attributes. The neuro-symbolic computing model (NN-PLP model) is more data-efficient than the NN model [32]. The PLP model learns the CVSS measurements data, and the output of the PLP model (based on logic reasoning) refines the output of the NN model (based on machine learning/deep learning). This study is the first study to combine logic reasoning and machine learning/deep learning in vulnerability risk prioritization to our best knowledge.

D. Integrating Threat Modeling Into ML/DL-Based Approach

Threat modeling aims to provide a systematic analysis of potential threats and vulnerabilities in a system. It is a process for capturing, organizing, and analyzing all the information that affects the security of a system [33]. Generally, there are two different threat modeling approachesin cybersecurity. The one approach is the most popular approach, which assumes that attackers can arbitrarily choose whichever attack vector or sequence that they think will maximize the function of the model assigned to them [34]. Based on this assumption, threat is modeled either from the perspective of vulnerability and technical exposure (e.g., attack graphs), or from the perspective of strategies (e.g., using game theory to model attacker strategies). Therefore, defenders need to defend against all vulnerabilities in the system [34]. For example, the attack graph enumerates all known vulnerabilities that attackers may exploit in a system. A threat model is a structured representation of such information [33]. According to the empirical observation, only a fraction of tens of thousands of possible vulnerabilities has been actively exploited in the wild [5]. Thus, this approach might not be efficient in remediation. The other approach is based on empirical views of attacks. By analyzing historical records of attacks worldwide, this approach assumes that attackers prefer to perform attacks that match their skills, goals, and capabilities.

The existing studyattempts to model threats from the attackers' views by analyzing the characteristics of highly exploited vulnerabilities [24]. When applying the threat model into risk assessment, attackers' motivation, capability, and the corresponding vulnerabilities are integrated to construct risk metrics [35]. Previous study [8] attempts to tune vulnerabilities' risk assessment by integrating the attacker model into the ML/DL-based model. It focuses on tuning vulnerabilities' assessment based on a known attacker group with machine learning models and finds a more accurate prediction of risk prioritization. Our solution also assumes that attackers have a preference on attacks in a network. Unlike [8], our solution is not limited to the known attacker groups. We extend our solution to attackers not covered in the reported known attacker groups by modeling threats from the historical threat records in a network and exploits in the wild (e.g., ExploitDB [36]). Additionally, we integrate the logic reasoning into the solution and develop an NN-PLP model to learn the identified threat attributes for a network system. In Section V, we compare LICALITY with these existing solutions (Expert-based solution [21] and ML/DL-based solution [8]), and show the results in Figures 9-10 and Tables VI-VII.

III. Motivation of LICALITY

A. Motivation of Integrating Threat Modeling in LICALITY

LICALITY creatively integrates threat modeling into vulnerability prioritization. The proposed threat modeling method has three arguments of threat (including historical threat and exploits record) as theoretical foundations: *Argument 1* **(A1)** the attackers' experiences on software services have attributes on assessing the likelihood of exploitation; *Argument 2* **(A2)** the vulnerabilities' access complexity and impact features have attributes on assessing the criticality of exploitation; and *Argument 3* **(A3)** the exploits records in the wild have attributes on assessing the likelihood of exploitation.

Attackers prefer to maximize their gains by leveraging costs and gains when using vulnerabilities in attacks [26]. The more difficult it is to exploit vulnerabilities, the higher costs attackers pay for; the more significant losses to victims when attacked, the higher gains attackers obtain. Thus, an attacker's strength (**A1**) on the experienced services/systems (e.g., the software set in the threat modeling method in Section IV-B) may reduce associated costs on learning services and result in increasing the likelihood of exploitation. Additionally, the CVSS vector of *Access Complexity* (AC) and three impact metrics (*Confidentiality Impact, Integrity Impact*, and *Availability Impact* (C,I,A)) works as an indicator for analyzing the exploits [24], [37]. The AC indicates the costs of attack since it measures how difficult it is to explore this vulnerability. The C,I,A impact metrics indicate the gains of attack since it measures how significant losses could be when the vulnerability is compromised. Thus, the CVSS $\langle AC, C, I, A\rangle$ vector has attributes on indicating the potential gains for attackers (e.g.,

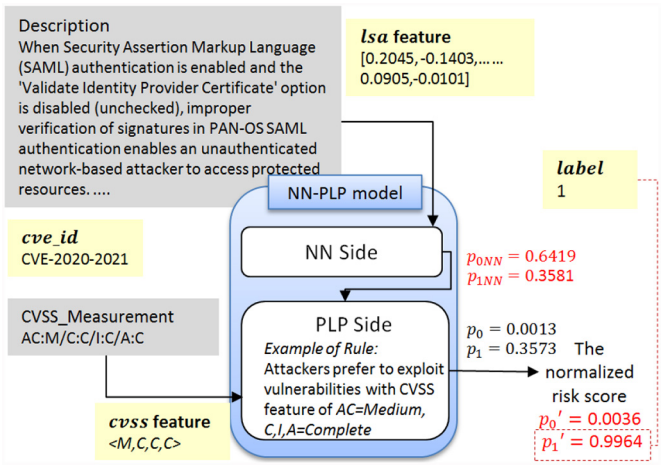


Fig. 1. A running example of assessing CVE-2020-2021 with LICALITY.

how severe/critical impact is on the defended network when exploiting the vulnerability), which represent the criticality of exploitation in the network (**A2**). Furthermore, the previous study [8] proves that the exploits record in the wild is the critical label for training the supervised risk models to assess how likely a vulnerability be exploited. Thus, the exploits record (**A3**) has attributes on assessing the likelihood of exploitation. To encode the threat attributes, we develop the threat modeling method in Section IV-B based on these three arguments. In theexample of Figure 1, the software service associated with the vulnerability CVE-2020-2021 is 'PAN-OS', and the CVSS $\langle AC, C, I, A\rangle$ vector has the value of $\langle M, C, C, C\rangle$.[1]

B. Motivation of Using the NN-PLP Model in LICALITY

Figure 1 shows a motivation example of using LICALITY to assess CVE-2020-2021. The existing ML/DL-based solution often begins by processing the vulnerability's data. The data processing tools (e.g., NLP tools) often represent the vulnerability in the form of a vector representation (e.g., *cve_id* CVE-2020-2021's *lsa* feature[2]). The *lsa* feature is learned by an ML/DL model (e.g., the NN model in Figure 1) to output $p_{1_{NN}} = 0.3581$ as a risk score. However, due to the possible reasons discussed in Section II-C, solely depending on the ML/DL-based model has limitations. In this motivation example, $p_{1_{NN}} = 0.3581$ is a weak estimation of the actual risk in reality. The vulnerability CVE-2020-2021 was reported in an advanced persistent threat (APT) attack by the Cybersecurity and Infrastructure Security Agency (CISA) [38]. Thus, the desired output should identify this vulnerability as risky.

To overcome these limitations, LICALITY develops the PLP model to efficiently learn threat attributes identified by threat modeling, e.g., attackers prefer to exploit *Medium* complexity (AC = Medium), *High* impact vulnerabilities (C,I,A = Complete). Thus, in the proposed NN-PLP model, the probabilistic program, which is associated with CVE-2020-2021's *cvss* feature $\langle M, C, C, C\rangle$, learns a higher probabilistic value in the PLP model. The PLP model refines the NN model's initial assessment as $p_1 = p_{1_{NN}} * p_{1_{PLP}} = 0.3573$ and $p_0 = p_{0_{NN}} * p_{0_{PLP}} = 0.0013$ (the computation is based on Equation (1) discussed in Section IV-C. Finally, LICALITY generates the normalized risk score, where $p_1' = 0.9964 > p_0' = 0.0036$,[3] which matches the desired output (*label* is 1). LICALITY's output more accurately represents this vulnerability's real risk, where is summarized in Table I.

TABLE I
COMPARING OUTPUTS FROM NN AND PLP FOR CVE-2020-2021 IN FIGURE 1

Models	The likelihood of exploitation from NN model	Considering the criticality of exploitation from PLP model
Outputs	$p_{0_{NN}} = 0.6419 > p_{1_{NN}} = 0.3581$	$p_0' = 0.0036 < p_1' = 0.9964$
Assessment	less risk	more risk

IV. SYSTEM AND MODEL OF LICALITY

This section presents the proposed study system, models, and assumptions of our proposed solution – LICALITY, a vulnerability risk prioritization model integrated with threat modeling. LICALITY addresses the vulnerability's prioritization issue based on the *Threat, Vulnerability, and Consequence* risk framework [39]. The *threat* is from the historical threat and exploits record in the wild (e.g., ExploitDB [36]). The *vulnerability* is identified by NVD [15], and the *consequence* is from CVSS metrics of vulnerability [3]. We utilize vulnerabilities' CVSS features identified by the expert-based approach of CVSS and the LSA features used in the ML/DL-based approach. Thus, our solution is built upon these two existing risk prioritization approaches by utilizing the novel neuro-symbolic computation. LICALITY assesses a vulnerability's risk from two perspectives as (1) *likelihood of exploitation*: a multi-source measurement considering the historical threat, exploits history (e.g., from ExploitDB), and vulnerability descriptions from security experts; and (2) *criticality of exploitation*: the access complexity and the impacts of successful exploitation based on CVSS measurements.

Figure 2 shows the overview of LICALITY's system architecture. LICALITY contains two components: neural network-probabilistic logic programming (NN-PLP)-based Learning (ⓐ) and vulnerability risk prioritization (ⓑ). Inthe NN-PLP-based learning (ⓐ) stage, LICALITY processes the NVD vulnerabilities to obtain the *lsa* feature and *cvss* feature of vulnerabilities through the vulnerability data processing (①). In the threat modeling (②), LICALITY labels the NVD vulnerabilities based on the identified threat attributes[4] and the labeling functions F_{labelA} and F_{labelB}. The processed LSA_feature, CVSS_feature, and label of vulnerabilities are used to construct a dataset (③). Additionally, LICALITY develops the probabilistic rules in the PLP model by the rule

[1] M/C: (M)edium, (C)omplete.

[2] The details of generating *lsa* feature and *cvss* feature are discussed in Section IV-A

[3] The sum of probabilities for a binary classifier outputted by the PLP model is not equal to 1. Thus we rank vulnerabilities based on a normalized output.

[4] The threat attributes are characteristics or distinguishing properties of a threat. The combined characteristics of a threat describe the threat's willingness and ability to pursue its goal [40].

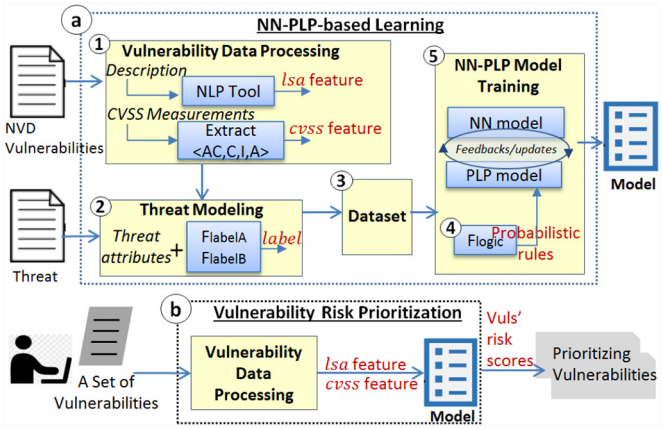


Fig. 2. The overview of LICALITY system architecture.

function F_{logic} (④) with the *cvss* feature in the dataset. Lastly, the developed NN-PLP model is trained (⑤) with the dataset to learn the encoded threat attributes. The learning process of the NN-PLP model is discussed in Section IV-C3.

In the vulnerability risk prioritization (ⓑ) stage, learning a model from the NN-PLP-based learning allows a security admin to evaluate a set of vulnerabilities' risk for future attacks. Given a set of vulnerabilities, LICALITY processes them through the vulnerability data processing, and generates the related *lsa* feature and the *cvss* feature. With the learned model, LICALITY assesses the likelihood of exploitation from *lsa* features, and assesses the criticality of exploitation from *cvss* features (discussed in Sections III-B and IV-C3). LICALITY consolidates the likelihood and criticality measurements to derive vulnerabilities' risk scores for future attacks (discussed as the future threat in this study in Section V).

A. Vulnerability Data Processing in LICALITY

As shown in step ① in Figure 2, the input of the *vulnerability data processing* function in LICALITY is a NVD dataset with vulnerabilities' description and CVSS measurements as shown in the gray box in Figure 1. The vulnerability data processing function extracts a vulnerability's cve_id, and extracts its *lsa* feature from description and *cvss* feature from CVSS measurements.

To extract the *lsa* features of vulnerabilities, a Natural Language Processing (NLP) tool of latent semantic analysis (LSA) [11] performs the data processing on text data of *Description*. LSA follows the statistical computation to generate the contextual-usage meaning of words to a large text corpus. The output of LSA focuses on representing the features hidden in the data. Such features are mentioned as "latent vulnerability features" in a previous study for risk prioritization [8]. Extracting the *lsa* features from text data usually takes three steps as follows:

1) Transform *Description* for all $Vul\_data \in NVD\_Vul$ into a corpus;
2) Use NLP's technique of Term Frequency-Inverse Document Frequency (TF-IDF) [11] assigns each identified term in the corpus a weight from 0 to 1.
3) Use truncated Singular Value Decomposition (SVD) algorithm [11] to extract *lsa* features from the TF-IDF matrix with a value between -1 and 1.

The TF-IDF weight indicates the importance of a term to adescription as a whole. TF-IDF weighs a term by calculating the product of TF and its IDF. The TF-IDF score shows how relevant a term is throughout all documents in a corpus. The truncated SVD algorithm finds the most valuable information of the data matrix. It reduces the TF-IDF matrix dimension by combining similar patterns between terms and documents into a latent feature vector with a value between -1 and 1 [11]. To extract *cvss* feature, a developed python script is used to obtain the value of $\langle AC, C, I, A\rangle$ vector for each vulnerability.

B. Threat Modeling in LICALITY

As we discussed in Section I, the existing risk prioritization approaches have limitations that the high CVSS base score does not realistically represent the actual exploits [5], [6]; and fora specific network's defense, the record of exploits in the wild [5], [8] does not associate to vulnerabilities that are more critical to the defended network [8]. According to the existing study [5], [6], the most relevant threat information should be considered in this risk model. Thus, we propose the threat modeling method based on the three arguments (A1,A2,A3). The threat modeling (shown in step ② in Figure 2) generates labels to a dataset to train the NN-PLP model. This method encodes the threat attributes (including the historical threat and the exploits record) through the labeling functions F_{labelA} and F_{labelB}. Given any vulnerability cve_id, Section IV-A shows how its *cvss* and *lsa* features are obtained. We define its *label* according to Definition 1 to construct a dataset (shown in step ③ in Figure 2).

Definition 1 (Threat Modeling): For $Threat = \{Historical\_Threat, Exploits\_Record\}$, threat attributes are identified by:

- *The risky vulnerability set* $Risky\_vul$*:* a set of vulnerabilities that are associated with the given historical threat;
- *The service set* S_{sw}*:* a set of software services that are associated with vulnerabilities in $Risky\_vul$;
- *The CVSS set* S_{cvss}*:* a set of $CVSS\_\langle AC, C, I, A\rangle$ that are associated to vulnerabilities in $Risky\_vul$;
- *The exploits record* $Exploits\_Record$*:* a set of vulnerabilities that are recorded in the Vulnerability Exploit Database (e.g., ExploitDB [36])

Based on threat attributes, the labeling function A and B are defined as:

- *Labeling function A* F_{labelA}*:*
  For a vulnerability with cve_id, if its description contains at least one software service in S_{sw}, or if its *cvss* feature is in S_{cvss}, then, its *label* is 1, otherwise, is 0.
- *Labeling function B* F_{labelB}*:*
  For a vulnerability with cve_id, if its cve_id is in $Exploits\_Record$, then, its *label* is 1, and is 0 otherwise.
- The logic OR is used to combine the labels generated by F_{labelA} and F_{labelB} as *label* for each vulnerability.

TABLE II
THE HISTORICAL THREAT OF CASE 1 AND CASE 2 IN THIS STUDY

The historical threat	Risky vulnerabilities ($Risky\_vul$)	The software set (S_{sw})	The CVSS set* (S_{cvss})
MVs_2015 [41] ($Case1$)	CVE-2006-3227, CVE-2008-2244, CVE-2009-3129, CVE-2009-3674, CVE-2010-0806, CVE-2010-3333, CVE-2011-0101, CVE-2012-0158, CVE-2012-1856, CVE-2012-4792, CVE-2013-0074, CVE-2013-1347, CVE-2014-0322, CVE-2014-1761, CVE-2014-1776, CVE-2014-4114	Microsoft Internet Explorer, Microsoft Office, Microsoft Word, Microsoft Excel, Open XML file format converter mac, SQL Server, Biztalk Server, Commerce Server, Visual Foxpro, Visual Basic, Host Integration Server, Microsoft Silverlight, Sharepoint Server, Microsoft Windows	$\langle L, C, C, C\rangle$, $\langle M, C, C, C\rangle$, $\langle H, N, P, N\rangle$
APTVs_2017 [42] ($Case2$)	CVE-2015-2590, CVE-2016-7255, CVE-2017-0263, CVE-2016-7855, CVE-2015-7645, CVE-2015-1701, CVE-2015-5119, CVE-2015-3043, CVE-2017-0262, CVE-2015-2424, CVE-2016-4117, CVE-2017-11292	Adobe flash, Java, Microsoft Windows, Microsoft office, Microsoft word	$\langle L, C, C, C\rangle$, $\langle L, P, P, P\rangle$, $\langle M, C, C, C\rangle$

*L/M/H/P/N/C: (L)ow, (M)edium, (H)igh, (P)artial, (N)one, (C)omplete.

We construct a dataset $\mathcal{D}$ in step ③ in Figure 2, where each data instance is a 4-tuple $\langle cve\_id, lsa, cvss, label\rangle$ where

- $cve\_id \in \mathbb{N}$ is the id of the vulnerability;
- $lsa \in \mathbb{R}^{150}$ is the vulnerability description's LSA feature;
- *cvss* is a 4-tuple $\langle ac, c, i, a\rangle$ as described in Definition 1;
- $label \in \{0, 1\}$ is the label of whether this vulnerability is less or more risky.

Table II shows the given historical threat of MVs_2015 and APTVs_2017 in this study in Section V, where the risky vulnerabilities ($Risky\_vul$), software set (S_{sw}), and CVSS set (S_{cvss}) are listed. These two historical threats are from existing cybersecurity reports [41], [42]. The details of these threats are discussed in Section V.

C. NN-PLP Model Training in LICALITY

We develop the NN-PLP model on a neuro-symbolic computation platform DeepProbLog [43]. As shown in Figure 1, the NN side takes LSA features, and the PLP side takes CVSS features as inputs. In the NN-PLP model, both sides treat each other as a black box. The NN-PLP model learns from the environment (on the neural network side), and reasons from what has been learned (on the reasoning side) [13]. During the training processes, the label is used to compute the loss to update the NN-PLP model's parameters.

1) The NN Side: The NN side is a neural network model with a gradient-based algorithm, such as, Adam [44], SGD [45], etc. The neural network can learn the vulnerability's *lsa* features, and output the risk score that is associated with the exploits in the previous study [8].

2) The PLP Side: LICALITY develops the PLP model in probabilistic logic programming (PLP) language [43], [46]. For a given data instance $\langle cve\_id, lsa, cvss, label\rangle$ in dataset $\mathcal{D}$, the PLP model contains three types of rules as probabilistic rule c_e (representing for the criticality of exploit) , the neural rule l_e (representing for the likelihood of exploit),and the logical rule *assessment*.

- *Logical rule*
  $assessment(cve\_id, ac, c, i, a, l) \leftarrow$
  $l\_e(lsa, l), c\_e(ac, c, i, a, l)$
- *Probabilistic rule*
  $p_{l_{PLP}}::c\_e(ac, c, i, a, l), l \in \{0, 1\}$
- *Neural rule*
  $nn(model\_net, lsa, l)::l\_e(lsa, l)$[5]

In the PLP model, l_e represents the NN model's output, where *lsa* is the *lsa* feature, *l* is *label*. The c_e represents the PLP model's output on the criticality of exploitation, where *ac*, *c*, *i*, *a* is the *cvss* feature. The probabilistic rules are developed by the rule function F_{logic} (shown in step ④ in Figure 2) defined as:

Definition 2 (Rule function F_{logic}): For all *cvss* features in $\mathcal{D}$, probabilistic rules are developed as:

$$p_{1_{PLP}}::c\_e(AC, C, I, A, 1),$$
$$p_{0_{PLP}}::c\_e(AC, C, I, A, 0),$$

where $p_{1_{PLP}}$ and $p_{0_{PLP}}$ are learned probabilities during training the NN-PLP model. Recall that c_e represents the PLP model's output on the criticality of exploitation. The *cvss* feature has several values for each $\langle AC, C, I, A\rangle$ measure, e.g., None, Partial, Complete, Low, Medium, and High. Thus, the total number of probabilistic rules is determined by $K = 2 \times$ *enum*(AC) $\times$ *enum*(C) $\times$ *enum*(I) $\times$ *enum*(A), where *enum*(X) is to enumerate the values of variable *X*.

3) Training the NN-PLP Model: During the NN-PLP-based learning (ⓐ), LICALITY trains the NN-PLP model by following the parameter learning of DeepProblog [43]. There are two types of parameter learning for the NN-PLP model in LICALITY as: 1) the NN model's parameters are updated by back-propagation [47] with the gradients of the loss w.r.t. the NN model's output (e.g., l_e);and 2) the PLP model's learnable parameters (e.g., $p_{1_{PLP}}$ and $p_{0_{PLP}}$ in Definition 2) are updated by gradient semiring [43] with the gradients of the loss w.r.t. the learnable probabilities in PLP model (e.g., in the c_e).The NN-PLP model learns parameters based on gradient descent by minimizing the loss function *L* as [43]:

$$\arg\min_{X} \frac{1}{|Q|} \sum_{(q,p)\in Q} L(P_X(q), p)$$

where *X* are the NN-PLP model's parameters. Given a set *Q* of pairs (*q*, *p*), *q* is a query of the logical program (e.g., *assessment*), *L* is the loss function, $P_X(q)$ is the query *q*'s probability with the given inputs (e.g., *lsa* feature and *cvss* feature), and *p* is the desired success probability (e.g., *label*).

[5]We use $p_{l_{NN}}$ to represent $nn(model\_net, lsa, l)$ for the purpose of presentation in Section IV-C3

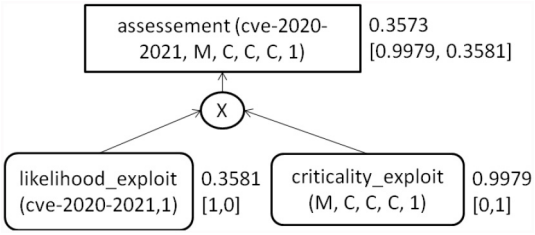


Fig. 3. The example of computing in SDD in LICALITY.

These gradients for parameter learning are computed in the Sentential Decision Diagram (SDD) [48]. The SDD is a tree structure that defines the logical and probabilistic relationships among nodes, and represents a propositional knowledge base for probabilistic reasoning. Refer to Figure 1, the input data instance for training the NN-PLP model is a 4-tuple with $\langle cve\_id, lsa, cvss, label\rangle$ shown in the yellow box.

Equation (1) [43] is used in the SDD to compute gradients of the loss w.r.t. the NN model's output and the learnable probabilities in PLP model, respectively.

$$\begin{aligned}&\left(p_{NN}, \overrightarrow{v_{p_{NN}}}\right) \bigotimes \left(p_{PLP}, \overrightarrow{v_{p_{PLP}}}\right)\\ &= \left(p_{NN}p_{PLP}, p_{PLP}\overrightarrow{v_{p_{NN}}} + p_{NN}\overrightarrow{v_{p_{PLP}}}\right)\end{aligned} \tag{1}$$

p_{NN} and p_{PLP} represent the probability outputted by the NN model and the PLP model's probabilistic rules in LICALITY, respectively. $\overrightarrow{v_{p_{NN}}}$ represents the partial derivative of p_{NN} w.r.t. the NN model's output. $\overrightarrow{v_{p_{PLP}}}$ represents the partial derivative of p_{PLP} w.r.t. the learnable probabilities in PLP model.

Figure 3 shows an example of these vectors ($\overrightarrow{v_{p_{NN}}}$ and $\overrightarrow{v_{p_{PLP}}}$) as [1, 0] (for l_e) and [0, 1] (for c_e).The formula $\overrightarrow{v_p} = p_{PLP}\overrightarrow{v_{p_{NN}}} + p_{NN}\overrightarrow{v_{p_{PLP}}}$ computes the gradients. For the NN model's output, the associated gradient is 0.9979; and for the PLP model's probabilistic rules, the associated gradient is 0.3581.

In addition, during the vulnerability risk prioritization ((b)), the input data is a 3-tuple $\langle cve\_id, lsa, cvss\rangle$. The *lsa* feature is fed into the NN model, and the *cvss* feature is fed into the PLP model. The probability of *assessment* is calculated by $p = p_{NN}p_{PLP}$ in Equation (1). For example, in Figure 3, the NN model (represented by the atom of $l\_e(l_{cve-2020-2021}, 1)$ in SDD)outputs the probability of $p_{1_{NN}} = 0.3581$, and the PLP model's probabilistic rule $c_e(M, C, C, C, 1)$has the probability of $p_{1_{PLP}} = 0.9979$. The assessed probability of $assessment(cve - 2020 - 2021, M, C, C, C, 1)$ is $p = p_{NN}p_{PLP} = 0.3573$. Similarly, the assessed probability of $assessment(cve - 2020 - 2021, M, C, C, C, 0)$ is 0.0013. Because the sum of probabilities for the predicate of *assessment* is not equal to 1. Therefore, we obtain the LICALITY's normalized risk score of 0.9964, where it is calculated by the following formula 0.3573/(0.3573 + 0.0013) = 0.9964.

V. Use LICALITY for Risk Prioritization

LICALITY proposes two key components: 1) the threat modeling method that identifies threat attributes of the given historical threat and exploits record, and 2) the Neuro-Symbolic model (the NN-PLP model) that learns such threat attributes. We present a comprehensive case study in this section to investigate LICALITY's performance (marked as *NN-PLP & Threat Modeling*) on risk prioritization with the existing approaches. The existing approaches are marked as *Existing Expert-based Solution* and *Existing ML/DL-based Solution* in Figure 9, Figure 10, Table VI, and Table VII. The *Existing Expert-based Solution* represents the predominant expert-based approach that leverages the CVSS scores [21] for risk prioritization. The *Existing ML/DL-based Solution* represents the existing ML/DL based approach that associates a risk metric of vulnerabilities with the existence of corresponding exploits under an attacker model [8].

Additionally, we investigate different AI techniques: a neuro-symbolic model (the NN-PLP model) and a neural network model (the NN-only model) to prioritize vulnerabilities for remediation.To compare the effect of the proposed threat modeling method, risks are identified as *Threat Modeling* and *Without Threat Modeling* (e.g., ExploitDB) in case studies. Furthermore, to investigate the best working scenario of LICALITY, the case studies cover different historical-future threat relationships[6] as:

- Case 1 (Microsoft Vulnerabilities (MVs)): the historical threat is from the high-risk Microsoft Vulnerabilities (called *MVs_2015* in this study) in 2015 CISA alert [41], and the future threat (called *MVs_2020*) is from the top routinely exploited Microsoft Vulnerabilities in 2020 CISA alert [49]. Microsoft became cyber attackers' preferred platform. The reported Microsoft vulnerabilities have risen 64% from 2015 to 2019 [50]. In Case 1, the historical threat and the future threat are associated with Microsoft products and share some same features (as shown in Table II and Table V).
- Case 2 (Advanced Persistent Threat Vulnerabilities (APTVs)): the historical threat (called *APTVs_2017*) is from the APT28 attacker in the FireEye 2017 report [42], and the future threat (called *APTVs_2020*) is from APT attack identified recently in 2020 CISA alert [38]. The APT attack is a stealthy threat that uses continuous and sophisticated hacking techniques to access a system [51]. Additionally, the APT attackers usually pursue their targets over months or years. Thus the APT attack is challenging to detect [51]. In Case 2, the historical threat and the future threat are both from the APT attack, and their features of threat modeling vary a lot (as shown in Table II and Table V).

A. Dataset Setup

Figure 4 shows an overview of the dataset structure in this study. We extract vulnerabilities from 1999 to 2021 June in NVD [15]. Some CVEs are rejected due to a duplicated record or are reserved for reports in the future. By excluding invalid CVEs marked as 'reject' or 'reserved' in descriptions or have

[6]The previous study revealed that there are some overlaps of features between the historical threat and future threat in a defended network by analyzing attacks in reality [5], [6]

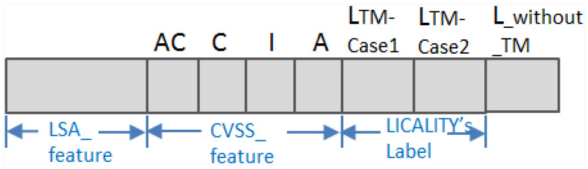


Fig. 4. Data structure of dataset for evaluation in this study.

empty CVSS records, we refine the NVD to 155,176 vulnerabilities in this study. We develop a JSON parser in Python to extract vulnerabilities' CVE_IDs, descriptions, and CVSS vector from NVD JSON files [15]. The CVSS features (e.g., $X_{i+1}, \ldots X_{i+4}$) and the LSA features (e.g., $X_0, \ldots, X_i$) are generated by using the vulnerability data processing function (defined in Section IV-A), where the NLP tool of LSA discovers features that cannot be directly measured in the data. LSA features are extracted from vulnerabilities in three steps:

1) Transform all 155,176 vulnerabilities' descriptions into a text corpus through data processing, including lowercasing all text data, stemming and transforming words into their root forms (e.g., using 'attack' to replace the words 'attacked', 'attacks'), removing stop-words (e.g., is, a the, have, etc.), normalizing words (e.g., using 'iptables' to replace 'ip-table', 'ip-tables', 'ip tables', etc.), and removing noise (e.g., digits characters, special symbols, etc.)
2) Transform the text data in corpus into the term frequency-inverse document frequency (TF-IDF) matrix through the NLP tool of TF-IDF [11].
3) Transform the TF-IDF matrix into LSA features through a truncated SVD algorithm [11]. We explore different features to generate the LSA features and select the best SVD explained variance ratio. In this study, the features number is 150, and the SVD explained variance ratio is 0.87. Such LSA features are shown as $(X_0, \ldots, X_i), i = 149$ in the dataset.

Three types of labels are associated with the features in the dataset as:

- *L_TM-Case1:* labels therisk of vulnerabilities by using LICALITY's threat modeling method with the identified historical threat of MVs_2015 and ExploitDB records in Case 1;
- *L_TM-Case2:* labels therisk of vulnerabilities by using LICALITY's threat modeling method with the identified historical threat of APTVs_2017 and ExploitDB records in Case 2;
- *L_without_TM:* labels therisk of vulnerabilities with the given exploits from ExploitDB without threat modeling.

For *L_TM-Case1* and *L_TM-Case2* , we firstly label the dataset by using the labeling function F_{labelA} (defined in Section IV-B) with the service set and the CVSS set of the historical threat. Then, we generate labels by using F_{labelB} with the exploits record from ExploitDB. Finally, a logic OR is performed to combine the labels generated by F_{labelA} and F_{labelB} as *L_TM-Case1* and *L_TM-Case2*, respectively. Each case study has one historical threat. In Case 1, the historical threat MVs_2015 is from the CISA alert [41], which advises IT and security professionals to prioritize patching these most commonly known vulnerabilities [14]. In Case 2, the historical threat APTVs_2017 is from the Cybersecurity company FireEye's advanced persistent threat (APT) report for the attackers' group APT28 [42]. There are 16 Microsoft vulnerabilities in MVs_2015 and 12 vulnerabilities in APTVs_2017. The threat modeling method labels the dataset based on the features of these identified vulnerabilities. For *L_without_TM*, we use the existence of exploits in ExploitDB [36] as the label, which is in the same way as the existing study [8], [20]. A vulnerability's *L_without_TM* is 1 if there is a corresponding exploit record in the ExploitDB; otherwise, is 0.

TABLE III
AN OVERVIEW OF DATASET

Property	Dataset
Vulnerability	NVD [15]
Exploits in the wild	ExploitDB [36]
Historical threat	MVs_2015 [41] (Case 1), APTVs_2017 [42] (Case 2)
Future threat	MVs_2020 [49] (Case 1), APTVs_2020 [38] (Case 2)
Positive labels	38,036 (*L_TM-Case1*), 65,871 (*L_TM-Case2*) , 11,679 (*L_without_TM*)

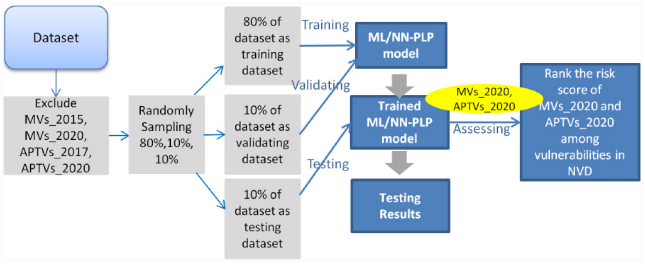


Fig. 5. An overview of separating the dataset.

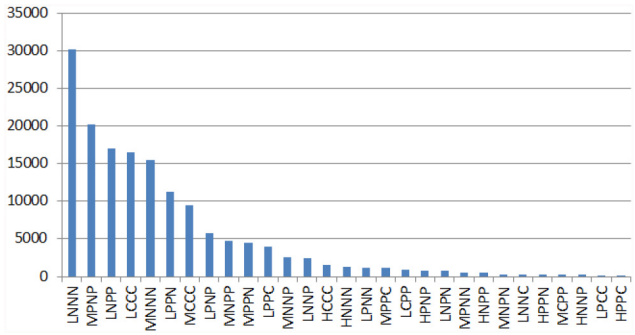


Fig. 6. An overview of CVSS features' distribution in the dataset.

Table III shows that there are 11,679 positive labels (labeled as 1) with *L_without_TM*, 38,036 positive labels with $L_{TM-Case1}$, and 65,871 positive labels with $L_{TM-Case2}$. In this study, the training set (contains 124,094 data), validation set (contains 15,513 data), and testing set (contains 15,511 data). These data sets are randomly sampled from 155,176 vulnerability data at the rate of 80%, 10%, 10%, respectively. Figure 5 shows the process of separating the dataset, where the identified historical threats and the future threats are all excluded from the training, validation, and testing set in this study. Figure 6 shows the distribution of the popular CVSS

features in the datasets, where each listed CVSS features in this figure are associated to over 100 vulnerabilities among all 155,576 vulnerabilities in the dataset. Figure 6 reveals that the CVSS features are not equally distributed. The CVSS features of $\langle L, N, N, N\rangle$, $\langle M, P, N, P\rangle$, $\langle L, N, P, P\rangle$, $\langle L, C, C, C\rangle$, $\langle M, N, N, N\rangle$, and $\langle L, P, P, N\rangle$ are associated to over 10,000 vulnerabilities in the dataset.In LICALITY, the PLP side of the NN-PLP model is trained with the CVSS features.

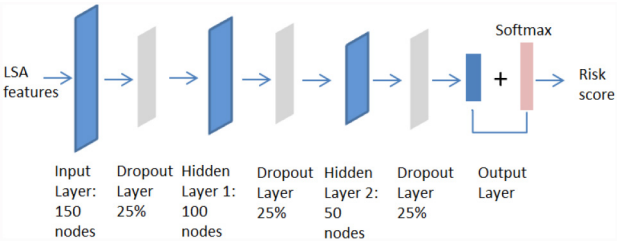


Fig. 7. The neural network model for assessing risk.

B. Performance Evaluation Design

Risk prioritization forms a basis for allocating resources, where the high-ranked vulnerabilities will be prioritized for taking defensive actions. The performance evaluation in this study contains two main measurements:

1) AUC (Area Under the receiver operating characteristics (ROC) Curve): the AUC is widely used to evaluate different classification models' performance with a single measurement. The existing study [8] uses this metric to evaluate the efficiency of machine learning algorithms as a classifier for a dataset in risk prioritization. In this study, the efficiency of using the NN-PLP model or NN-only model as a classifier for the dataset is evaluated by AUC.
2) The recommend rankings (percentage) of the assessed vulnerability among all vulnerabilities in the dataset: the existing study [8] uses this measurement to evaluate the performance of risk prioritization solutions. We compare the recommend rankings (percentage) of the future threats (e.g., MVs_2020 in Case 1 and APTVs_2020 in Case 2) among all vulnerabilities in the dataset under LICALITY (marked as *Threat Modeling*) and the existing approaches (marked as *ExploitDB* and *CVSS Score*) in Table VII.

The ROC curve is a plot that summarizes the performance of a binary classification model. The AUC represents the area under the ROC curve. It ranges from 0 to 1. If the model's predictions are 100% correct, it has an AUC of 1.0. Its x-axis indicates the false positive rate (FPR), and the y-axis shows the true positive rate (TPR). The FPR is computed as FPR = False Positives/(False Positives + True Negatives), and the TPR is computed as TPR = True Positives/(True Positives + False Negatives).

C. The Model Setup

In this section, we demonstrate the model setup in this study. There are two types of models in this study:

- *NN-only model:* A neural network model with a gradient-based algorithm. We follow the existing study [8] on latent feature-based risk prioritization and develop the neural network model in Pytorch.
- *NN-PLP model:* A neuro-symbolic model with neural network (NN) and probabilistic logic programming (PLP) techniques. To better compare the performance of these two models, we keep the NN-PLP model's NN side (e.g., neural network structure and the initial parameter settings) the same as the NN-only model.

The NN-only model uses the artificial neural network for predictive modeling. It has been used widely in binary classification. Figure 7 shows a four-layer neural network model developed in this study, which has 150 nodes in the input layer, 100 nodes in the first hidden layer, 50 nodes in the second hidden layer, the output layer has a softmax to normalize the output. The dropout layer is applied after both the input and hidden layers with a 0.25 dropout rate. The dropout layer can reduce overfitting on the training data by randomly dropping 25% of nodes' connections from a layer. The design of this neural network follows the previous study [8], where the neural network has more potential to be an accurate classifier than others for risk prioritization since it has the ability of batch training and multiple hidden layers. This paper mainly focuses on illustrating the proposed solution for risk prioritization, so investigating more NN model structure/parameters could be discussed in future work.

The NN-PLP model keeps its NN side the same as the NN-only model. The PLP side has probabilistic rules, neural rules, and logical relation rules. We use a vulnerability CVE-2020-2021 in Figure 3 as an example to illustrate the model setup on the PLP side:

- The probabilistic rules represent the vulnerability's CVSS features $\langle M, C, C, C\rangle$ in the PLP model, and are generated by the rule function F_{logic} defined in Section IV-B as:

$$p_{0_{PLP}} :: c\_e(M, C, C, C, 0),$$
$$p_{1_{PLP}} :: c\_e(M, C, C, C, 1),$$

  where $p_{0_{PLP}}$ and $p_{1_{PLP}}$ are the probabilities of the probabilistic rules, $p_{0_{PLP}} + p_{1_{PLP}} = 1$. The value of $p_{0_{PLP}}$ and $p_{1_{PLP}}$ are learned by training the NN-PLP model. By enumerating all possible values of $\langle AC, C, I, A\rangle$, 162 unique probabilistic rules are in the PLP model.
- The neural rule represents the output of the NN model as: $nn(net, lsa, [0, 1]) :: l\_e(lsa, 0),\ l\_e(lsa, 1)$, where for the vulnerability CVE-2020-2021, the neural network *net* takes its *lsa* features as inputs, and then outputs the probabilities of $l\_e(lsa, 0)$ and $l\_e(lsa, 1)$.
- The logical rules represent the assessment of risk, which are measured by both the likelihood and the criticality of exploitation. The logical rules for the vulnerability CVE-2020-2021 are as:
  $assessment(cve - 2020 - 2021, M, C, C, C, 0) \leftarrow$
  $l\_e(lsa, 0),\ c\_e(M, C, C, C, 0);$
  $assessment(cve - 2020 - 2021, M, C, C, C, 1) \leftarrow$
  $l\_e(lsa, 1),\ c\_e(M, C, C, C, 1),$

where *assessment*(*cve*-2020-2021, *M*, *C*, *C*, *C*, 1) represents the assessed risk for CVE-2020-2021 when it is exploited in the defended network system. Because the sum of *assessment*(*cve*-2020-2021, *M*, *C*, *C*, *C*, 1) and *assessment*(*cve*-2020-2021, *M*, *C*, *C*, *C*,0) are not equal to 1, the assessed risk score of vulnerability CVE-2020-2021 is the normalized value of *assessment*(*cve*-2020-2021, *M*, *C*, *C*, *C*,1).

D. Comparative Study Results and Analysis

In the comparative study, we attempt to compare the efficiency of using the neural network model (NN-only model) with using the neuro-symbolic model (NN-PLP model) on prioritizing vulnerability based on risks identified by *Threat Modeling*. The results show the effectiveness of using the NN-PLP model to learn the risk associated with the threat modeling method. We also investigate the best working scenario of LICALITY by assessing the future threats in two cases.

1) Training the NN-Only Model: To train the NN-only model, the *lsa* features are fed into the model, and a validation loss is calculated on the validation set at the end of each training epoch. The inputs are the *lsa* features space described by a Jaccard similarity matrix on vulnerabilities' description. Early stopping framework [52] is introduced into the model to monitor the validation loss, and to stop the training when no improvement is observed. Early stopping is widely used in Pytorch to avoid overfitting. The model with minimal validation loss is selected as the trained model. We feed the selected trained model with the testing set to obtain the test result.

The NN-only model shown in Figure 7 applies the Multilayer Perceptron (MLP) algorithm. In this study, we also investigate another gradient-based algorithm (Logistic Regression) covered in the previous study on learning LSA features [8]. The logistic regression algorithm performs a regression analysis on a linear relationship between the predictor variables and the events' logits (a logarithm of probabilities). A log softmax is applied on the output layer with Criterion = NLLLoss and Optimizer = SGD [45]. By following the previous study [8], both MLP and logistic regression algorithms are used to learn the LSA features. The MLP (AUC = 0.877) and logistic regression (AUC = 0.875) are very close, and MLP has better performance on the recommended ranking of assessed future threat in Case 1 and Case 2. Thus, this study focuses on usingthe MLP algorithm. Additionally, we also investigate the optimizer of Adam [44]. Since the SGD performs better on the testing set, we select SGD as the optimizer of the NN-only model in this study. Nonetheless, future work can explore a more complicated NN model. In this study, we focus on demonstrating the LICALITY solution, not on developing the NN model. The AUC of MLP and logistic regression in this study both are better than what was reported in the previous study [8], so the NN-only model is well trained for *lsa* features in this study and is a valid comparative model for LICALITY.

2) Training NN-PLP Model: The NN-PLP model is constructed on the platform of DeepProbLog [43]. Thus the

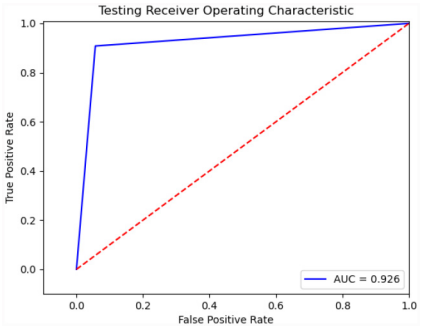


Fig. 8. The ROC curve of LICALITY (NN-PLP model) in Case 2.

TABLE IV
COMPARING THE NN-ONLY MODEL AND THE NN-PLP MODEL ON CLASSIFYING RISK IDENTIFIED BY *Threat Modeling* IN THIS STUDY

Models	Parameters	AUC $L_{TM-Case1}$	AUC $L_{TM-Case2}$
NN-only Model	Criterion = BCELoss, Optimizer=SGD	0.844	0.844
NN-PLP Model	Criterion = BCELoss, Optimizer=SGD	0.846	0.926

way of learning parameters on the NN model and PLP model follows the design of DeepProbLog. The inputs are the vulnerabilities' *lsa* features and *cvss* features. The dataset label $L_{TM-Case1}$ and $L_{TM-Case2}$ represents a vulnerability-exploit relationship identified by the threat attributes in two cases. Figure 6 shows the *cvss* features in the dataset. The *cvss* features are fed into the NN-PLP model to adjust the parameter of probabilistic atoms in the probabilistic rules. We train the NN-PLP model with a gradient-descent-based optimizer as SGD. Table IV shows the trained NN-PLP model and the NN-only model's performance, where the NN-PLP's AUC of Case 2 is 0.926 as shown in Figure 8. The high AUC indicates that the trained NN-PLP models can classify very well on the testing set than the NN-only model for the risk identified by threat modeling.

3) Vulnerability Risk Prioritization With the Trained Model in Case 1 and Case 2: In Case 1, we prioritize 7 Microsoft vulnerabilities of the MVs_2020 threat identified by a CISA alert in 2020 [49]. As shown in Table V, there are 5 Microsoft software services and 2 different $\langle AC, C, I, A\rangle$ vectors as $\langle M, C, C, C\rangle$ and $\langle L, P, P, P\rangle$ in CVSSv2. In Case 2, we prioritize 8 vulnerabilities of APTVs_2020 threat identified by a CISA alert in 2020 [38]. This threat covers 8 software services and performs vulnerability chaining among them. There are 5 different $\langle AC, C, I, A\rangle$ vectors associated with these vulnerabilities in CVSSv2. We also compare the changes of $\langle AC, C, I, A\rangle$ vectors inCVSSv3, where the *Assess Complexity* (AC) in CVSSv2 is separated into the *Attack Complexity* and the *User Interaction* in CVSSv3. In CVSSv3, *Attack Complexity* is measured as Low and High, *User Interaction* is measured as None and Required, and $\langle C, I, A\rangle$ are measured as None, Low, and High.

TABLE V
THE SOFTWARE SERVICES AND CVSS VECTOR OF THE ASSESSED VULNERABILITIES (FUTURE THREAT) IN CASE 1 AND CASE 2

Case	CVE ID	The software service	The CVSS(v2) feature [1] $\langle AC, C, I, A\rangle$	The CVSS(v3) feature [1] $\langle AC, C, I, A\rangle$
Case 1	CVE-2012-0158	Microsoft Office	$\langle M, C, C, C\rangle$	N/A
	CVE-2015-1641	Microsoft Word	$\langle M, C, C, C\rangle$	N/A
	CVE-2017-0143	Microsoft Windows	$\langle M, C, C, C\rangle$	$\langle H, N, H, H, H\rangle$
	CVE-2017-0199	Microsoft Office	$\langle M, C, C, C\rangle$	$\langle L, R, H, H, H\rangle$
	CVE-2017-8759	Microsoft NET	$\langle M, C, C, C\rangle$	$\langle L, R, H, H, H\rangle$
	CVE-2017-11882	Microsoft Office	$\langle M, C, C, C\rangle$	$\langle L, R, H, H, H\rangle$
	CVE-2019-0604	Microsoft SharePoint	$\langle L, P, P, P\rangle$	$\langle L, N, H, H, H\rangle$
Case 2	CVE-2018-13379	FortiOS	$\langle L, P, N, N\rangle$	$\langle L, N, H, H, H\rangle$
	CVE-2019-11510	Pulse Connect Secure, Policy Secure	$\langle L, P, P, P\rangle$	$\langle L, N, H, H, H\rangle$
	CVE-2019-19781	Citrix Controller, Gateway, SDWAN WANOP	$\langle L, P, P, P\rangle$	$\langle L, N, H, H, H\rangle$
	CVE-2020-1472	Microsoft Windows Server	$\langle M, C, C, C\rangle$	$\langle L, N, H, H, H\rangle$
	CVE-2020-1631	Junos OS	$\langle M, P, P, P\rangle$	$\langle L, N, H, H, H\rangle$
	CVE-2020-2021	PAN-OS	$\langle M, C, C, C\rangle$	$\langle L, N, H, H, H\rangle$
	CVE-2020-5902	BIG-IP devices	$\langle L, C, C, C\rangle$	$\langle L, N, H, H, H\rangle$
	CVE-2020-15505	MobileIron Core, Connector	$\langle L, P, P, P\rangle$	$\langle L, N, H, H, H\rangle$

[1]CVSSv2 $\langle (A)ccess(C)omplexity\rangle$: (L)ow, (M)edium, (H)igh; $\langle C, I, A\rangle$: (N)one,(P)artial, (C)omplete. [2]CVSSv3 $\langle (A)ttack(C)omplexity\rangle$: (L)ow, (H)igh; $\langle (U)ser(I)nteraction\rangle$: (N)one, (R)equired; $\langle C, I, A\rangle$: (N)one, (L)ow, (H)igh. N/A: Not available.

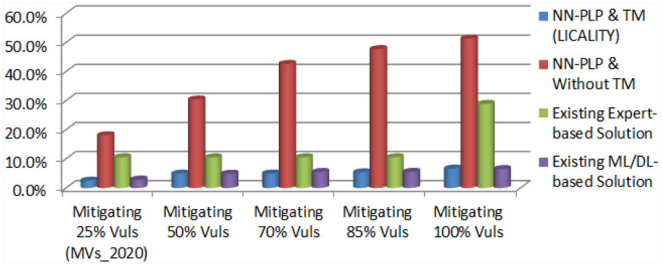


Fig. 9. Comparing the risk prioritization rankings (in % of all vulnerabilities) of top exploited Microsoft vulnerabilities (MVs_2020) in Case 1.

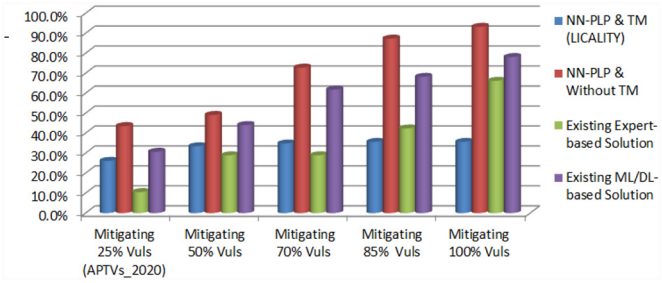


Fig. 10. Comparing the risk prioritization rankings (in % of all vulnerabilities) of APT attacker chained vulnerabilities (APTVs_2020) in Case 2.

We compute the percentages of MVs_2020 and APTVs_2020's vulnerabilities that rank among all 155,176 vulnerabilities in the dataset. Viewing ranked vulnerabilities in descending order, the higher the risk score, the lower the percentage is shown in Table VII. A vulnerability with the lowest percentage will first be recommended for remediation. We also compute these percentages for *Existing Expert-based Solution* and *Existing ML/DL-based Solution*. The *Existing Expert-based Solution* ranks vulnerabilities based on the CVSS score [21].The *Existing ML/DL-based Solution* follows the previous study [8] that integrates attacker model into risk prioritization.

Figure 9 and Figure 10 compare the risk rankings of the assessed vulnerabilities in two cases, where the y-axis represents the risk rankings among all 155,176 NVD vulnerabilities. The x-axis represents the number of vulnerabilities in the future threat (MVs_2020 or APTVs_2020) recommended for mitigation. A lower bar height in Figure 9 and Figure 10 leads to a higher priority for the recommendation among all vulnerabilities in the dataset, which means that a smaller amount of remediation work is associated with mitigation.

In Case 1, Figure 9 shows that LICALITY outperforms the *Existing Expert-based Solution [21]* and *NN-PLP & Without threat modeling* on correctly ranking MVs_2020 on the top position (with the lower bar height shown in Figure 9) for mitigation. Additionally, LICALITY has slightly better performance than the *Existing ML/DL-based Solution [8]* on classifying the risk identified by *Threat Modeling* ($L_{TM-Case1}$) in Case 1. Table VI shows the details of Figure 9. For example, if remediating 70% of

TABLE VI
SUMMARY OF PERCENTAGES OF VULNERABILITIES IN TABLE VII THAT WOULD BE RECOMMENDED TO MITIGATE INCREASING PERCENTAGES OF VULNERABILITIES IN CASE 1 AND CASE 2

Case	The Mitigated Vulnerabilities	Existing Expert-based Solution [21]%	Existing ML/DL-based Solution [8]%	LICALITY%
Case 1	70%	18.5755	5.4963	4.9736
	85%	18.5755	5.5504	5.3714
	100%	19.2563	6.5087	6.6536
Case 2	70%	28.8986	61.7424	34.7993
	85%	42.3362	68.1686	35.5897
	100%	66.1836	78.1082	35.6226

TABLE VII
COMPARING RISK SCORES AND RANKING (%) OF MVS_2020 AND APTVS_2020 WITH THE PROPOSED NN-PLP MODEL AND THE EXISTING METHOD

Case	CVE ID	Existing Expert-based Solution [21]	Existing ML/DL-based Solution [8]	NN-PLP & Without Threat Modeling	NN-PLP & Threat Modeling (LICALITY)	Existing Expert-based Solution [21]%	Existing ML/DL-based Solution [8]%	NN-PLP & Without Threat Modeling %	NN-PLP & Threat Modeling (LICALITY)%
Case 1	CVE-2012-0158	9.3	0.6678	0.0328	0.9999	10.5215	6.5087	18.1268	5.3714
	CVE-2015-1641	9.3	0.7994	0.0166	0.9999	10.5215	2.8632	30.4388	3.8990
	CVE-2017-0143	9.3	0.6998	0.0911	0.9999	10.5215	5.5504	8.7585	4.9736
	CVE-2017-0199	9.3	0.9141	0.0299	1.0000	10.5215	0.8835	19.4458	1.3499
	CVE-2017-8759	9.3	0.7979	0.0067	0.9999	10.5215	2.8896	51.3403	4.9175
	CVE-2017-11882	9.3	0.7006	0.0077	0.9999	10.5215	5.4963	47.7456	6.6536
	CVE-2019-0604	7.5	0.7220	0.0096*	1.0000*	28.8986	4.8777	42.6024	2.5342
Case 2	CVE-2018-13379	5	0.0891	0.0124*	0.9981*	66.1836	78.1082	36.8017	34.7993
	CVE-2019-11510	7.5	0.1477	0.0084*	0.9988*	28.8986	68.1686	45.6362	33.7156
	CVE-2019-19781	7.5	0.1862	0.0092*	0.9991*	28.8986	61.7424	43.4701	32.7841
	CVE-2020-1472	9.3	0.6304	0.0073	0.9989	10.5215	30.6701	49.0730	33.3630
	CVE-2020-15505	7.5	0.8728	0.0043*	0.9999*	28.8986	12.5663	62.7999	11.8929
	CVE-2020-1631	6.8	0.3707	0.0010*	0.9965*	42.3362	44.7238	93.1684	35.5897
	CVE-2020-2021	9.3	0.3828	0.0029	0.9964	10.5215	44.0239	72.8433	35.6226
	CVE-2020-5902	10	0.5616	0.0015	0.9997	5.4499	34.3492	87.2562	26.0750

* by updating $\langle C, I, A\rangle$ with the CVSSv3 vector in TableV.

Microsoft vulnerabilities in Table VII, LICALITY correctly places them on the top 4.9736% of all 155,176 vulnerabilities, while the existing expert-based solution [21] places them on the top 18.5755%. The comparative results show that LICALITY (with NN-PLP and threat modeling method) correctly places these seven vulnerabilities near the top of the recommendation list for remediation. In Case 1, LICALITY reduces the vulnerability remediation work required by the *Existing Expert-based Solution [21]* by a factor of 2.89 (over 19K vulnerabilities, i.e., calculated as $155176*(19.2563\% - 6.6536\%) = 19556$), and obtains almost the same performance as the *Existing ML/DL-based Solution [8]*. In Case 2, Figure 10 shows that LICALITY outperforms the existing two solutions on vulnerability risk prioritization. As shown in Table VI, when remediating 100% vulnerabilities of APTVs_2020, LICALITY reduces the vulnerability remediation work required by the *Existing Expert-based Solution [21]* by a factor of 1.85 (over 47K vulnerabilities), and reduces such remediation work required by the *Existing ML/DL-based Solution [8]* by a factor of 2.19 (over 65K vulnerabilities, i.e., calculated as $155176*(66.1836\% - 35.6226\%) = 47423$; $155176*(78.1082\% - 35.6226\%) = 65927$).

Details of the risk scoring and ranking data are shown in Table VII, where the risk scores and the associated rankings are the updated assessments by updating the $\langle C, I, A\rangle$ to CVSSv3.

E. Discussion

Our work is a new attempt to improve the existing risk prioritization solutions by assessing vulnerability risk from the likelihood of exploitation and the criticality of exploitation. A threat modeling method is proposed to encode a historical threat's attributes into the dataset. The NN-PLP model is trained to learn such threat attributes.This study investigates the best working scenario of LICALITY by comparing it to the corresponding risk ranking results in two cases. The LICALITY (NN-PLP model & threat modeling method) successfully prioritizes all vulnerabilities of the future threats on the top positions. Although the research outcomes are promising, several subtle issues need to be discussed.

1) Threat Modeling: The *Threat modeling* method encodes the historical threat's attributes as labels in the dataset and as logic rules in the PLP side of the NN-PLP model. Based on our best knowledge, Alperin *et al.*'s work [8] is the first study to encode an attacker model's (APT28) target software service into risk prioritization through labeling. In the evaluation, we compare LICAILITY with this study that is identified as *Existing ML/DL-based Solution [8]* in Section V). LICALITY can reach the almost same performance in Case 1, and significantly outperform this existing solution in Case 2. Additionally, their work only focuses on the known attacker (e.g., labels with APT28's attributes, and then prioritizes APT28's vulnerabilities). LICALITY has no requirements on the attack signatures of attackers.

In this study, LICALITY (NN-PLP model) has a better AUC than the NN-only model, which indicates that the NN-PLP model is an effective classifier for the risk identified through the threat modeling method. Moreover, the historical threat (MVs_2015 and APTVs_2017) is published before the future threat (MVs_2020 and APTVs_2020). LICALITY correctly places the high-risk vulnerabilities of the future threat on the top positions, which outperforms the existing approaches. These findings indicate that LICALITY assesses vulnerabilities based on the defended network's historical threat and can prioritize newly discovered vulnerabilities.

Furthermore, by comparing the results of risk prioritization in two cases, LICALITY reduces more vulnerabilities in Case 1 than in Case 2. This finding matches our expectation since these two cases cover different historical-future threat

relationships and have different attack detection difficulties. Case 2 is a more challenging working scenario for LICALITY since the future threat APTVs_2020 contains vulnerabilities mostly published in 2020. The software services vary a lot compared to what in the historical threat APTVs_2017, so the historical threat' attributes on the attacker's strength and preference (encoded as the software set (S_{sw})) might be hard to support assessing this future threat. Additionally, the APT attack is a stealthy threat that uses continuous and sophisticated hacking techniques to gain access to a system, which requires a more in-depth analysis on vulnerabilities' interactions in vulnerability chaining [38]. Although being evaluated under this more challenging case, LICALITY prioritizes all vulnerabilities in the top 35.6226%, which reduces the vulnerability remediation work required by the *Existing Expert-based Solution [21]* by a factor of 1.85, and reduces such remediation work required by the *Existing ML/DL-based Solution [8]* by a factor of 2.19.

2) Reasoning in the NN-PLP Model: In this study, the logical reasoning in the NN-PLP model is based on the most straightforward correlation relation. The NN side's weights are adjusted based on a given historical threat's attributes (e.g., the risky vulnerabilities ($Risky\_vul$), the software set (S_{sw}), and the CVSS set (S_{cvss})) through the learning iterations. It will be interesting to incorporate a system environment setup into the reasoning model with more complicated reasoning relations.

3) Impacts on CVSS Measurement: CVSSv3 is the current version of CVSS, including base score, temporal, and environmental metrics. The machine learning/deep learning-based approach (e.g., using LSA features as input to assess vulnerability's risk [8]) can serve as an excellent supplement to the existing CVSS measurement that mainly focuses on evaluating the severity/criticality of vulnerabilities. LICALITY reasons on the likelihood of exploitation from LSA features and criticality of exploitation from experts' knowledge (CVSS features). This way, it can serve as a novel approach to enhance the rigid equations given in the existing CVSS measurement model.

4) How to Incorporate the Use of the Solution for a Real System: Due to the lack of exploited vulnerability data sources from a real system, in this study, we emulate a real system by considering Microsoft vulnerabilities in MVs_2015 and APT attack vulnerabilities in APTVs_2017 as historical threats to perform threat modeling for a given network system. The overall result is promising and shows improvements in risk prioritization. It may not have all historical threats from the given dataset for a given system, and in reality, it can be very challenging to build such a dataset. Thus, it will be interesting to deploy LICALITY in a specific network and simulate the real work scenario for vulnerability management to test its performance.

5) Evaluation Issues: Another subtle issue we noticed in this study (shown in Table VII) is that CVSSv2 does not correctly measure the impact vector $\langle C, I, A\rangle$ in some cases. After updating the corresponding value to CVSSv3, LICALITY well prioritizes vulnerability. Also, we only use one-step reasoning rules in LICALITY. More sophisticated multi-step reasoning rules should be considered for complicated scenarios byconsidering computer network topology, firewall rules, defense strategies, etc., into the threat model to improve the assessment accuracy.

VI. Conclusion

In this study, we design LICALITY, a risk prioritization system, using the threat modeling method to encode threat attributes in a dataset. LICALITY uses a neuro-symbolic approach with the NN-PLP model to learn these threat attributes. LICALITY provides a realistic assessment solution based on a given network's past risk/attack history (refer to the historical threat in this study). The comparative case study demonstrates how to apply LICALITY and shows that LICALITY can enhance the use of CVSS by providing more practical evaluation results for decision-makers to prioritize their actions in responding to newly discovered vulnerabilities (refer to the future threat in this study). Next, we plan to keep investigating this solution's effectiveness under a more realistic scenario (e.g., considering computer network topology, firewall rules, or other threat intelligence information) in our future study.

Appendix

APT:Advanced Persistent Threat
APTVs: Advanced Persistent Threat Vulnerabilities *
$\langle AC, C, I, A\rangle$: $\langle AccessComplexity, ConfidentialityImpact, IntegrityImpact, AvailabilityImpact\rangle$*
AUC: Area Under Curve
CVE: Common Vulnerabilities and Exposures
CVSS: Common Vulnerability Scoring System
CISA: Cybersecurity and Infrastructure Security Agency
DL:Deep Learning
FPR: False Positive Rate
LICALITY: the name of the proposed solution that represents the likelihood and criticality of exploitation *
LSA: Latent Semantic Analysis
MLP: Multilayer Perceptron
ML: Machine Learning
MVs: Microsoft Vulnerabilities *
NIST: National Institute of Standards and Technology
NN: Neural Network
NVD: National Vulnerability Database
PLP: Probabilistic Logic Programming
ROC curve: Receiver Operating Characteristic Curve
TM: Threat Modeling *
TPR: True Positive Rate
SDD: Sentential Decision Diagram
SGD: Stochastic Gradient Descent
* The acronyms defined in this study.

Acknowledgment

The authors thank Michael Xin and the anonymous reviewers for their valuable comments and suggestions. All the authors are gratefully thankful for research grants from Naval Research Lab N0017319-1-G002 and National Science Foundation U.S. DGE-1723440 and OAC-2126291.

REFERENCES

[1] *Risk Impact Assessment and Prioritization*, The Mitre Corporat., McLean, VA, USA, Accessed: Feb. 1, 2021. [Online]. Available: https://www.mitre.org/publications/systems-engineering-guide/acquisition-systems-engineering/risk-management/risk-impact-assessment-and-prioritization

[2] J. Oltsik, "Vulnerability Management Woes Continue but There Is Hope." 2019. [Online]. Available: https://www.esg-global.com/blog/vulnerability-management-woes-continue-but-there-is-hope (Accessed: Apr. 5, 2021).

[3] Forum of Incident Response and Security Teams, Cary, NC, USA *Common Vulnerability Scoring System SIG*. (2021). Accessed: Jul. 1, 2021. [Online]. Available: https://www.first.org/cvss/

[4] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system," *IEEE Security Privacy*, vol. 4, no. 6, pp. 85–89, Nov. 2006.

[5] L. Allodi and F. Massacci, "Comparing vulnerability severity and exploits using case-control studies," *ACM Trans. Inf. Syst. Secur. (TISSEC)*, vol. 17, no. 1, pp. 1–20, 2014.

[6] J. Jacobs, S. Romanosky, I. Adjerid, and W. Baker, "Improving vulnerability remediation through better exploit prediction," *J. Cybersecur.*, vol. 6, no. 1, 2020, Art. no. tyaa015.

[7] P. Johnson, "What Is CVSS v3.1? Understanding the New CVSS." (2019). [Online]. Available: https://resources.whitesourcesoftware.com/blog-whitesource/understanding-cvss-v3–1n (Accessed Feb. 1, 2021).

[8] K. Alperin, A. Wollaber, D. M. Ross, P. C. Trepagnier, and L. Leonard, "Risk prioritization by leveraging latent vulnerability features in a contested environment," in *Proc. 12th ACM Workshop Artif. Intell. Secur.*, 2019, pp. 49–57.

[9] C. Sabottke, O. Suciu, and T. Dumitras, "Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits," in *Proc. 24th {USENIX} Secur. Symp.*, 2015, pp. 1041–1056.

[10] B. L. Bullough, A. K. Yanchenko, C. L. Smith, and J. R. Zipkin, "Predicting exploitation of disclosed software vulnerabilities using open-source data," in *Proc. 3rd ACM Int. Workshop Secur. Privacy Anal.*, 2017, pp. 45–53.

[11] S. T. Dumais, "Latent semantic analysis," *Ann. Rev. Inf. Sci. Technol.*, vol. 38, no. 1, pp. 188–230, 2004.

[12] L. De Raedt, R. Manhaeve, S. Dumancic, T. Demeester, and A. Kimmig, "Neuro-symbolic = Neural + Logical + Probabilistic," in *Proc. NeSy@ IJCAI 14th Int. Workshop Neural Symbolic Learn. Reason.*, 2019, pp. 7–10.

[13] A. D. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran, "Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning," 2019, *arXiv:1905.06088*.

[14] The Cybersecurity and Infrastructure Security Agency, Arlington, VA, USA, *About CISA*. (2021). Accessed: Dec. 9, 2020. [Online]. Available: https://www.cisa.gov/about-cisa

[15] The National Institute of Standards and Technology, Gaithersburg, MD, USA, *NVD: National Vulnerability Database*. (2020). Accessed: Aug. 25, 2020. [Online]. Available: https://nvd.nist.gov/general

[16] R. A. Miura-Ko and N. Bambos, "SecureRank: A risk-based vulnerability management scheme for computing infrastructures," in *Proc. IEEE Int. Conf. Commun.*, 2007, pp. 1455–1460.

[17] A. Younis, Y. K. Malaiya, and I. Ray, "Assessing vulnerability exploitability risk using software properties," *Softw. Qual. J.*, vol. 24, no. 1, pp. 159–202, 2016.

[18] Q. Liu, Y. Zhang, Y. Kong, and Q. Wu, "Improving VRSS-based vulnerability prioritization using analytic hierarchy process," *J. Syst. Softw.*, vol. 85, no. 8, pp. 1699–1708, 2012.

[19] G. Yadav, P. Gauravaram, and A. K. Jindal, "SmartPatch: A patch prioritization framework for SCADA chain in smart grid," in *Proc. 26th Annu. Int. Conf. Mobile Comput. Netw.*, 2020, pp. 1–3.

[20] M. Edkrantz, S. Truvé, and A. Said, "Predicting vulnerability exploits in the wild," in *Proc. IEEE 2nd Int. Conf. Cyber Secur. Cloud Comput.*, 2015, pp. 513–514.

[21] P. Mell, K. Scarfone, and S. Romanosky, "A complete guide to the common vulnerability scoring system version 2.0," in *Forum Incident Resp. Security Teams*, vol. 1, Cary, NC, USA, Jul. 2007, p. 23.

[22] S. Figueroa-Lorenzo, J. Añorga, and S. Arrizabalaga, "A survey of IIoT protocols: A measure of vulnerability risk analysis based on CVSS," *ACM Comput. Surveys*, vol. 53, no. 2, pp. 1–53, 2020.

[23] A. Feutrill, D. Ranathunga, Y. Yarom, and M. Roughan, "The effect of common vulnerability scoring system metrics on vulnerability exploit delay," in *Proc. 6th Int. Symp. Comput. Netw. (CANDAR)*, 2018, pp. 1–10.

[24] L. Allodi and F. Massacci, "Attack potential in impact and complexity," in *Proc. 12th Int. Conf. Availability Rel. Secur.*, 2017, pp. 1–6.

[25] T. Zimmermann, N. Nagappan, and L. Williams, "Searching for a needle in a haystack: Predicting security vulnerabilities for windows vista," in *Proc. 3rd Int. Conf. Softw. Test. Verification Validation*, 2010, pp. 421–428.

[26] L. Allodi, F. Massacci, and J. Williams, "The work-averse cyberattacker model: Theory and evidence From two million attack signatures," *Risk Anal.*, to be published.

[27] C. Eiram and B. Martin, "The CVSSv2 Shortcomings, Faults, and Failures Formulation." [Online]. Available: https://www.riskbasedsecurity.com/reports/CVSS-ShortcomingsFaultsandFailures.pdf (Accessed: Mar. 1, 2021.)

[28] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond heuristics: Learning to classify vulnerabilities and predict exploits," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2010, pp. 105–114.

[29] H. Chen, R. Liu, N. Park, and V. Subrahmanian, "Using twitter to predict when vulnerabilities will be exploited," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2019, pp. 3143–3152.

[30] K.-W. Chang, S.-W. Yih, and C. Meek, "Multi-relational latent semantic analysis," in *Proc. Conf. Empir. Methods Natural Lang. Process. (EMNLP)*, Oct. 2013, pp. 1602–1612. [Online]. Available: https://www.microsoft.com/en-us/research/publication/multi-relational-latent-semantic-analysis/

[31] N. Sun, J. Zhang, P. Rimba, S. Gao, L. Y. Zhang, and Y. Xiang, "Data-driven cybersecurity incident prediction: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1744–1772, 2nd Quart., 2019.

[32] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," 2019, *arXiv:1904.12584*.

[33] *Threat Modeling*, The Open Web Application Security Project, Bel Air, MD, USA, Accessed: Mar. 3, 2021. [Online]. Available: https://owasp.org/www-community/Threat_Modeling

[34] L. Allodi and S. Etalle, "Towards realistic threat modeling: Attack commodification, irrelevant vulnerabilities, and unrealistic assumptions," in *Proc. Workshop Autom. Decis. Making Active Cyber Defense*, 2017, pp. 23–26.

[35] S. Bromander, A. Jøsang, and M. Eian, "Semantic Cyberthreat Modelling," in *Proc. STIDS*, 2016, pp. 74–78.

[36] *Exploitdb*, The Offensive Security, New York, NY, USA, (2020). Accessed: Aug. 25, 2020. [Online]. Available: https://www.exploit-db.com/

[37] D. M. Ross, A. B. Wollaber, and P. C. Trepagnier, "Latent feature vulnerability ranking of CVSS vectors," in *Proc. Summer Simulat. Multi-Conf.*, 2017, pp. 1–12.

[38] *APT Actors Chaining Vulnerabilities Against SLTT, Critical Infrastructure, and Elections Organizations*, The Cybersecurity & Infrastructure Security Agency (CISA), Arlington, VA, USA,, (2020). Accessed: Jul. 1, 2021. [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/aa20–283a

[39] S. Kaplan and B. J. Garrick, "On the quantitative definition of risk," *Risk Anal.*, vol. 1, no. 1, pp. 11–27, 1981.

[40] M. Mateski *et al.*, "Cyber threat metrics," U.S. Dept. Energy, Sandia Nat. Lab., Albuquerque, NM, USA, Tech. Rep. SAND2012-2427, 2012.

[41] *Top 30 Targeted High Risk Vulnerabilities*, The Cybersecurity & Infrastructure Security Agency (CISA), Arlington, VA, USA, (2015). Accessed: Mar. 7, 2021. [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/

[42] *APT28: At the Center of the Storm*, The FireEye, Milpitas, CA, USA, (2017). Accessed: Jul. 1, 2021. [Online]. Available: https://www2.fireeye.com/rs/848-DID-242/images/APT28-Center-of-Storm-2017.pdf

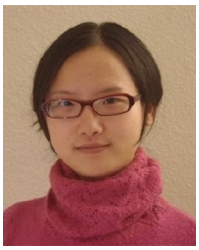
[43] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt, "Deepproblog: Neural Probabilistic Logic Programming," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3749–3759.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

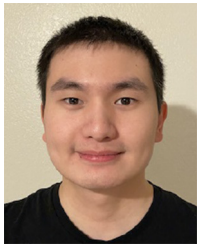
[45] *Stochastic Descent*, Wiki, San Francisco, CA, USA, Accessed: Apr. 4, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Stochastic_gradient_descent

[46] L. De Raedt, A. Kimmig, and H. Toivonen, "ProbLog: A probabilistic prolog and its application in link discovery." in *Proc. IJCAI*, vol. 7, 2007, pp. 2462–2467.

[47] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
[48] A. Darwiche, "SDD: A new canonical representation of propositional knowledge bases," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, 2011, pp. 819–826.
[49] *Top 10 Most Exploited Vulnerabilities 2016–2019*, The Cybersecurity and Infrastructure Security Agency, Arlington, VA, USA, (2020). Accessed: Aug. 3, 2020. [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/
[50] *Microsoft Vulnerabilities Report 2020*, Microsoft, Albuquerque, NM, USA, (2020). Accessed: Mar. 8, 2021. [Online]. Available: https://www.beyondtrust.com/resources/whitepapers/microsoft-vulnerability-report
[51] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2nd Quart., 2019.
[52] *Early Stopping*, Pytorch, (2021). Accessed: Jul. 1, 2021. [Online]. Available: https://pytorch-lightning.readthedocs.io/en/latest/common/early_stopping.html



Zhen Zeng (Graduate Student Member, IEEE) received the B.S. degree from Xidian University, China, and the M.S. degree from Purdue University, West Lafayette, IL, USA. She is currently pursuing the Ph.D. degree in computer science with Arizona State University. She worked in the high-tech industry for many years. Her research interests include vulnerability management, network and information security, and AI in cybersecurity.



Zhun Yang received the B.S. degree in information security from Wuhan University and the M.S. degree in computer engineering from Arizona State University, where he is currently pursuing the Ph.D. degree in computer science with the Automated Reasoning Group. His research focuses on the integration of neural networks and symbolic logic where neural network inference and training are enhanced by knowledge and automated reasoning.

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

Chun-Jen Chung received the Ph.D. degree in computer science from Arizona State University in 2015. He was a Software Developer with Microsoft and Oracle for several years, and a co-founder of a startup company providing cloud-base online education service. He is a Senior Architect with the Cloud Lab, Futurewei Technology Inc. His research interests are cybersecurity, software-defined networking, large scale, and high performance cloud infrastructure.