Risk-based Network Vulnerability Prioritization

by

Zhen Zeng

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved May 2022 by the
Graduate Supervisory Committee:

Guoliang Xue, Chair
Huan Liu
Ming Zhao
Yezhou Yang

ARIZONA STATE UNIVERSITY

August 2022

ABSTRACT

This dissertation investigates the problem of efficiently and effectively prioritizing a vulnerability risk in a computer networking system. Vulnerability prioritization is one of the most challenging issues in vulnerability management, which affects allocating preventive and defensive resources in a computer networking system. Due to the large number of identified vulnerabilities, it is very challenging to remediate them all in a timely fashion. Thus, an efficient and effective vulnerability prioritization framework is required. To deal with this challenge, this dissertation proposes a novel risk-based vulnerability prioritization framework that integrates the recent artificial intelligence techniques (i.e., neuro-symbolic computing and logic reasoning). The proposed work enhances the vulnerability management process by prioritizing vulnerabilities with high risk by refining the initial risk assessment with the network constraints. This dissertation is organized as follows. The first part of this dissertation presents the overview of the proposed risk-based vulnerability prioritization framework, which contains two stages. The second part of the dissertation investigates vulnerability risk features in a computer networking system. The third part proposes the first stage of this framework, a vulnerability risk assessment model. The proposed assessment model captures the pattern of vulnerability risk features to provide a more comprehensive risk assessment for a vulnerability. The fourth part proposes the second stage of this framework, a vulnerability prioritization reasoning engine. This reasoning engine derives network constraints from interactions between vulnerabilities and network environment elements based on network and system setups. This proposed framework assesses a vulnerability in a computer networking system based on its actual security impact by refining the initial risk assessment with the network constraints.

To my beloved family, my grandma, mom and dad, father-in-law, husband, and kids for their endless support and love.

ACKNOWLEGEMENTS

This dissertation would have been impossible without the help from my Ph.D. advisors Dr. Dijiang Huang and Dr. Guoliang Xue. I would like to give my special thanks to them for opening the door of Cybersecurity to me, guiding me on my Ph.D. journey, and offering me great freedom in exploring various interesting research topics throughout my years in the Computer Science program from cybersecurity education, attack analysis, to vulnerability management.

I would like to thank my committee members, Dr. Huan Liu, Dr. Ming Zhao, and Dr. Yezhou Yang, for valuable suggestions and insightful comments on my dissertation. I greatly appreciate the committee members for their questions during my comprehensive exam and proposal defense, which guided me to explore more deeply and broadly for my dissertation. I would like to thank Dr. Sharon Hsiao and Dr. Chun-Jen Chung for the invaluable discussion and collaborations.

I truly appreciate the time working with a diverse group of co-authors and lab members. I was lucky to work with Dr. Yuli Deng, Zhun Yang, Dr. Duo Lu, Dr. Ankur Chowdhary, Dr. Sowmya Myneni, Neha Vadnere, Kritshekhar Jha, Abdulhakim Sabur, and Garima Agrawal.

I want to thank my grandma, parents and father-in-law for their love and support. I also want to thank my kids, although they are too little to understand my research, they believe what I have done is COOL. I could not complete my Ph.D. journey without the support from my husband, who always believes in me and encourages me to become the best version of myself. He is my lover, best friend and partner.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Vulnerability management can enhance the computer networking system security by identifying, evaluating, and mitigating vulnerabilities in a computer networking system. Vulnerability prioritization is a critical procedure in the vulnerability management. There are 43% of cybersecurity professionals report that vulnerability prioritization is one of the most challenging issues in vulnerability management (Oltsik, 2019). In practice, vulnerability management contains three main steps. First, cyber admins identify the vulnerabilities in a computer networking system. Secondly, the identified vulnerabilities are prioritized based on risk levels. Lastly, cyber admins remediate the system by mitigating or removing high-risk vulnerabilities. Prioritizing vulnerabilities is required for efficiently managing the numerous vulnerabilities in a computer networking system (Dempsey *et al.*, 2020; Aboud, 2019).

1.1 Challenges of Vulnerability Prioritization in a Computer Networking System

The common vulnerability scoring system (CVSS) (The Forum of Incident Response and Security Teams, 2021; Mell *et al.*, 2006) has been widely used for vulnerability prioritization. The CVSS metrics contain three groups, i.e., base, temporal and environmental metrics, where the temporal and environmental metrics are optional. The CVSS score is derived from a static formula that depends on the experts' knowledge. Using high CVSS base scores alone in vulnerability prioritization has limitations, since the high CVSS score usually does not realistically represent the actual vulnerability exploits (Allodi and Massacci, 2014; Jacobs *et al.*, 2020). In practice, there are 13.5% vulnerabilities (over 15k) in national vulnerability database (NVD) with the highest CVSS base scores range (9-10) (Alperin *et al.*, 2019), and only 10-15% publicly known vulnerabilities have a known exploit, and even fewer are weaponized as part of hacking tool-kits (Jacobs *et al.*, 2020). 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 (Allodi and Massacci, 2014)." The CVSS is developed "to measure the severity of a vulnerability and should not be used alone to assess risk (Johnson, 2019)." Additionally, the CVSS method has limitations on handling such a large amount of vulnerabilities in a timely fashion. In practice, there are thousands or more vulnerabilities which can be discovered in an enterprise network (Alperin *et al.*, 2019). More new security vulnerabilities are disclosed in 2020 than ever before, with an average rate of 50 vulnerabilities per day (Woollacott, 2021). However, according to the existing CVSS scoring schema, it is infeasible to easily implement the CVSS temporal and environmental metrics to adjust a CVSS base score w.r.t. a real system's setup in practice. To implement these two metrics usually requires a security expert understanding both temporal and environmental situations very well, which leads to difficulty in practical application (Jack Wallen, 2020; Nikolai Mansourov, 2018), especially for the system with limited time and resources.

As the development of artificial intelligence (AI) techniques, the machine learning-based models support vulnerability prioritization by learning the patterns of vulnerability exploits. Researchers attempt to use machine learning (ML)/deep learning (DL)-based approaches to explore better associations between vulnerability features and exploits (Bozorgi *et al.*, 2010; Chen *et al.*, 2019). The existence of exploits is used as a significantly better risk metric than CVSS scores (Alperin *et al.*, 2019; Allodi and Massacci, 2014; Jacobs *et al.*, 2020; Sabottke *et al.*, 2015). However, solely depending on the supervised ML/DL-based model for risk prioritization still has limitations. The existing ML/DL-based approaches assess risk by estimating how vulnerabilities' features are associated with the exploits in the wild (Alperin *et al.*, 2019; Allodi and Massacci, 2014; Sabottke *et al.*, 2015; Jacobs *et al.*, 2020), where such association varies on different datasets and ML/DL methods (Bullough *et al.*, 2017). The previous study also shows that the selection of training and testing dataset critically affects the estimated performance on assessing the likelihood of vulnerability exploitation (Bullough *et al.*, 2017).

It is worth mentioning that regarding a vulnerability risk in a computer networking system, the security impact of a vulnerability is also closely associated with interactions between a vulnerability and network environment elements (Chung *et al.*, 2013; Lai and Hsia, 2007; Ou *et al.*, 2005). To analyze vulnerability-based attack scenarios in a network, the existing solutions (Ou *et al.*, 2005; Lai and Hsia, 2007) enumerate all possible vulnerability-based attack paths to a given attacking target as an attack graph (Ou *et al.*, 2005). The attack graph-based approach is not specifically designed for vulnerability prioritization. Therefore, when applying this approach on solving vulnerability prioritization problems, it has some limitations in practice. One attack graph only fits to a specific attack model with the "start" point and the "target" point. In practice, for an attack model with a remote attacker, every host in the network should be evaluated as both the "start" and "target" node (Lai and Hsia, 2007). When using the attack graph-based model to generate such attack graph for each host in a large scale network, it usually has high complexity, which hinders security admins in adopting this approach for vulnerability prioritization. Additionally, some organizations prioritize vulnerabilities simply depending on the asset value of a machine. The asset value can indicate the most vital assets to an organization, but cannot indicate the risks that directly affect this machine. Such risks might come from threats or vulnerabilities in a network. From the perspective of vulnerability management, it is more efficient to remediate the high-risk vulnerabilities that actually affect the assets in a computer networking system than patching all vulnerabilities associated with these high-value assets.

Furthermore, the existing study attempts to assess a vulnerability from various perspectives, such as, attacker behavior and system criticality (Yadav and Paul, 2019), vulnerability performance metrics (Farris *et al.*, 2018), network architecture and potential node interactions (Miura-Ko and Bambos, 2007), the attack path derived from the vulnerabilities dependency graph (Duan *et al.*, 2019), and so on. These solutions either assume vulnerabilities have equal effects on network nodes (Miura-Ko and Bambos, 2007; Farris *et al.*, 2018) or fail to address the network environment into the assessment model (Duan *et al.*, 2019). None of these existing solutions can adopt vulnerability prioritization to the network environment changes.

1.2 Research Problems

To address the aforementioned challenges, this dissertation focuses on the development of a novel network risk-based vulnerability prioritization framework that leverages multiple vulnerability risk features and network and security constraints to assess the overall risk of a vulnerability to a computer networking system.

Research Problems. This dissertation investigates the two types of research problems below:

- What are the risk features that determine a vulnerability's security impact in a computer networking system?

- How does a framework effectively and efficiently perform vulnerability risk prioritization in a computer networking system?

To address these research problems, the main contributions of the dissertation are summarized as follows. I characterize the risk features of a vulnerability in a computer networking system based on the cybersecurity risk model. Then, I propose a novel risk-based vulnerability prioritization framework. This proposed framework contains two stages. The first stage of this framework is a proposed vulnerability risk assessment model that utilizes the neuro-symbolic computing technique to learn these risk features. The second stage of this framework is a proposed vulnerability prioritization reasoning engine, which utilizes logic reasoning techniques to refine the risk assessment model with network constraints in a computer networking system. The rest of the dissertation is organized as follows. In Chapter 2, I present the literature review of the related work. In Chapter 3, I propose the risk-based vulnerability prioritization framework. In Chapter 4, I characterize the vulnerability risk features. In Chapter 5, I propose the vulnerability risk assessment model that is capable of learning the characterized risk features. In Chapter 6, I develop the prioritization reasoning engine that refines the risk assessment model by reasoning with network constraints. In Chapter 7, I conclude this dissertation and discuss the future work.

Chapter 2

LITERATURE REVIEW

According to the Verizon 2021 data breach investigations report, less than 25% of discovered vulnerabilities can be remediated (i.e., be fixed or be patched) within 30 days (Verizon Corp., 2021). 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 (Rapid7 Corp., 2020); and (b) defenders may want to postpone remediation due to management issues, such as regular schedule of software release/upgrade, to avoid downtime that is required to patch a vulnerability (Rapid7 Corp., 2020). The existing studies that support vulnerability prioritization in a network system are grouped as follows.

The Expert-based Approaches The existing predominant vulnerability prioritization approach is the CVSS scoring schema (Mell *et al.*, 2006, 2007). The CVSS is based on a static formula and cannot automatically associate vulnerabilities with network environmental elements (Spring *et al.*, 2018; Shah *et al.*, 2019). The CVSS score is widely used in the expert-based risk prioritization. The CVSSv2 was launched in 2007 (Mell *et al.*, 2007), 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 $CVSS_{base} = Impact \times Exploitability$ (Allodi and Massacci, 2014). The CVSSv2 base score defines the severity level of a vulnerability as Low (0-3.9), Medium (4-6.9), and High (7-10). The CVSSv3 (released in 2015) and CVSSv3.1 (released in 2019) change several metrics and values, as shown in Figure 2.1. It defines the severity level as None (0), Low (0.1-3.9), Medium (4.0-6.9), High (7.0-8.9), and Critical (9.0-10.0) (The Forum of Incident Response and Security Teams, 2021). CVSSv3 separates the *Access Complexity (AC)* in the CVSSv2 base score metric into *Attack Complexity* (to address an attack's former condition) and *User Interaction (UI)* (to address its latter conditions) (The Forum of Incident Response and Security Teams, 2021). The Confidentiality, Integrity, and Availability impact metric values also change from CVSSv2 as None, Partial, and Complete to CVSSv3 as None, Low, and High. The value in CVSSv2 reflects the overall percentage of the systems impacted by an attack, and the value in CVSSv3 represents the overall degree of impact the attack causes (The Forum of Incident Response and Security Teams, 2021). Since the CVSSv3 or CVSSv3.1 were released very recently, many vulnerabilities have been determined only for CVSSv2 (Figueroa-Lorenzo *et al.*, 2020).

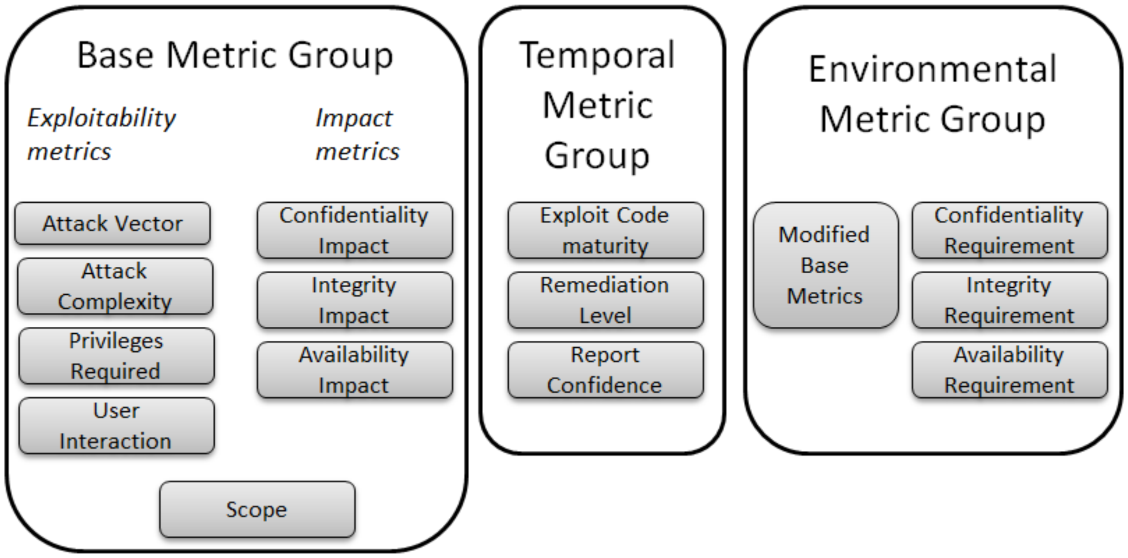


Figure 2.1: The CVSSv3 Metrics Group (The Forum of Incident Response and Security Teams, 2015)

Although the CVSS score does not correlate well with attacks in the wild (Alperin *et al.*, 2019), the previous study still agrees that the CVSS provides an excellent estimation of vulnerabilities' criticality (Feutrill *et al.*, 2018; Allodi and Massacci, 2017; Zimmermann *et al.*, 2010; Allodi *et al.*, 2021). In practice, organizations and companies make their changes to the CVSSv2 scheme to make it work better in their working scenarios. 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 (Allodi and Massacci, 2017). 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 (Allodi and Massacci, 2017), and a trade-off in vulnerability's impact and complexity measurements (Allodi *et al.*, 2021). For example, the work-averse cyber attackers prefer to exploit Low complexity High impact vulnerabilities (Allodi *et al.*, 2021). Researchers also study the effect of different CVSS measurements on vulnerability exploit delay, where the CVSS measure is used as a condition to filter data (Feutrill *et al.*, 2018). Additionally, 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 (Nikolai Mansourov, 2018).

The AI-based Approaches Researchers attempt to explore better associations between vulnerabilities' features and exploits (Bozorgi *et al.*, 2010; Chen *et al.*, 2019) by using the machine learning/deep learning (ML/DL) model. The key risk factors of vulnerabilities contain the probability of compromise, consequence, and time (Alperin *et al.*, 2019). The existence of exploits is identified as a significantly better risk metric than CVSS scores (Alperin *et al.*, 2019; Sabottke *et al.*, 2015; Allodi and Massacci, 2014; Jacobs *et al.*, 2020). Multiple databases (e.g., ExploitDB, *Symantec* WINE, NVD, and Twitter data) are used to build a dataset for training and testing machine learning models (Sabottke *et al.*, 2015; Bozorgi *et al.*, 2010; Chen *et al.*, 2019; Chang *et al.*, 2013). The NVD stores vulnerability data and is maintained by the National Institute of Standards and Technology (NIST). The vulnerabilities in NVD have the unique Common Vulnerabilities and Exposures (CVE) identification number, 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 (Jacobs *et al.*, 2020; Alperin *et al.*, 2019; Allodi and Massacci, 2014); and (2) the extracted vulnerabilities' features vary among studies (Sabottke *et al.*, 2015; Bullough *et al.*, 2017; Alperin *et al.*, 2019). The attack/non-attack data are usually highly imbalanced, and the selection of training and testing datasets and critically affects the estimated performance of the predictive model (Bullough *et al.*, 2017); and (3) the limitation of tools for processing vulnerability 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 (Alperin *et al.*, 2019; Sabottke *et al.*, 2015)) might be hard to capture the features of vocabularies that emerged later with high quality (Sun *et al.*, 2018). 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.

Threat modeling is introduced into vulnerability prioritization to enhance the ML-based model by the recent study (Alperin *et al.*, 2019). Threat modeling provides 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 (The Open Web Application Security Project, 2021). Usually, the threat modeling method assumes that attackers can arbitrarily choose whichever attack vector or sequence that they think will maximize the function of the model assigned to them (Allodi and Etalle, 2017). 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 (Allodi and Etalle, 2017). For example, an attack graph enumerates all known vulnerabilities that attackers may exploit in a system. A threat model is a structured representation of such information (The Open Web Application Security Project, 2021). According to the empirical observation, only a fraction of tens of thousands of possible vulnerabilities has been actively exploited in the wild (Allodi and Massacci, 2014). Thus, this approach might not be efficient in vulnerability remediation.

The existing study attempts to model threats from the attackers' views by analyzing the characteristics of highly exploited vulnerabilities (Allodi and Massacci, 2017). When applying the threat model into risk assessment, attackers' motivation, capability, and the corresponding vulnerabilities are integrated to construct risk metrics (Bromander *et al.*, 2016). A previous study attempts to tune vulnerabilities' risk assessment by integrating the attacker model into the ML/DL-based model (Alperin *et al.*, 2019). It tunes vulnerabilities' assessment based on a known attacker group with ML models and finds a more accurate prediction of risk prioritization. But this existing study is limited to the known attacker groups.

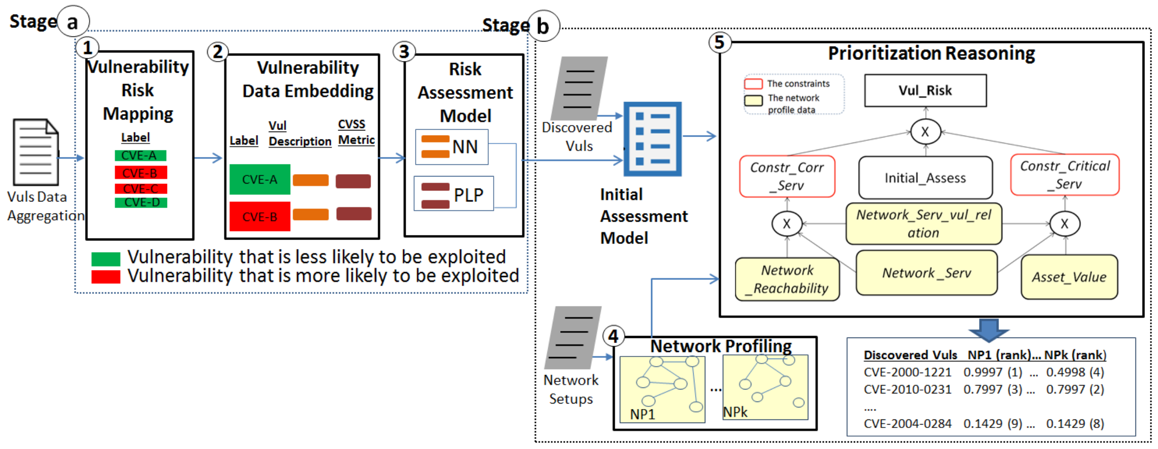
Network Specific Approaches The attack graph-based approaches (Tupper and Zincir-Heywood, 2008; McQueen *et al.*, 2006; Lai and Hsia, 2007; Jajodia *et al.*, 2005) aggregate the risk assessments for all vulnerabilities in the network as a single value to represent the system-level security, which cannot prioritize vulnerability individually. To analyze an attack scenario, all possible paths to a given attacking target in the network are enumerated (Ou *et al.*, 2005; Sheyner *et al.*, 2002; Lai and Hsia, 2007), which are impractical in practice. To enumerate all possible attack "start" points and "target" points in these previous work would be very difficult in a large-scale network with hundreds or thousands of machines.

Chapter 3

PROPOSED RISK-BASED VULNERABILITY PRIORITIZATION FRAMEWORK

This dissertation proposes a risk-based vulnerability prioritization framework. In this chapter, an overview of this proposed framework is presented. This framework assesses a vulnerability risk based on the characterized vulnerability risk features (discussed in Chapter 4). This framework contains two stages, including the vulnerability risk assessment (discussed in Chapter 5) and the vulnerability prioritization (discussed in Chapter 6).

3.1 Proposed Framework Overview



Discovered Vuls	NP1 (rank)...	NPk (rank)
CVE-2000-1221	0.9997 (1) ...	0.4998 (4)
CVE-2010-0231	0.7997 (3) ...	0.7997 (2)
....		
CVE-2004-0284	0.1429 (9) ...	0.1429 (8)

Figure 3.1: The Overview of the Proposed Risk-based Vulnerability Prioritization Framework

In this section, the overview system architecture of the proposed risk-based vulnerability prioritization framework is presented in Figure 3.1. In this framework, a vulnerability risk is assessed based on the characterized vulnerability risk features in Chapter 4, including *the likelihood of exploitation*: a multi-source measurement considering vulnerability exploitation history, vulnerability descriptions, access complexity, and network reachability of a host that has this vulnerability; and *the criticality of exploitation*: a multi-source measurement considering the security impact of successful vulnerability exploits. Finally, this risk assessment is refined by *the network constraints* in the prioritization reasoning engine, which prioritizes vulnerabilities under the network and security constraints in a computer networking system. These two stages are illustrated as follows.

Stage ⓐ Vulnerability Risk Assessment (discussed as LICALITY in Chapter 5) processes historical security data of a network to extract the threat attributes in this network and learns such threat attributes by a neuro-symbolic learning-based assessment model. In this stage, the risk assessment of vulnerability is based on the likelihood and criticality measurements of vulnerability exploitation.

Stage ⓑ Vulnerability Prioritization (discussed as ILLATION in Chapter 6) incorporates the actual network and security setups (i.e., network reconfiguration, access control, firewall rules updates, etc.) as the network profile to derive the network constraints and then consolidates these constraints and the initial risk assessment derived from the previous stage. ILLATION's output can help cyber admins understand the overall risk of a vulnerability w.r.t. the network profile. In this way, cyber admins can evaluate different network profiles to assess the effectiveness of different vulnerability mitigation strategies.

In the stage ⓐ *Vulnerability Risk Assessment*, ① *Vulnerability-Risk Mapping* function maps NVD vulnerabilities to threat attributes. Such threat attributes are identified from the threat in the *Vulnerability Data Aggregation*. The ② *Vulnerability Data Embedding* function embeds vulnerabilities as two types of features, which are extracted from vulnerability descriptions and CVSS metrics. The ③ *Assessment Model Training* function learns the identified threat attributes from the embedded data. In the stage ⓑ *Vulnerability Prioritization* (discussed as ILLATION in Chapter 6), a cyber analyst is able to prioritize the discovered vulnerabilities under various network profiles that are associated with mitigation actions. In this way, ILLATION could help cybersecurity admins prioritize vulnerabilities under different attack assumptions. The ④ *Network Profiling* embeds the given network profiles as a knowledge base, and the ⑤ *Prioritization Reasoning* function generates the network constraints and then prioritizes vulnerabilities by consolidating the network constraints and the initial risk assessment derived by the vulnerability risk assessment model.

3.2 Summary

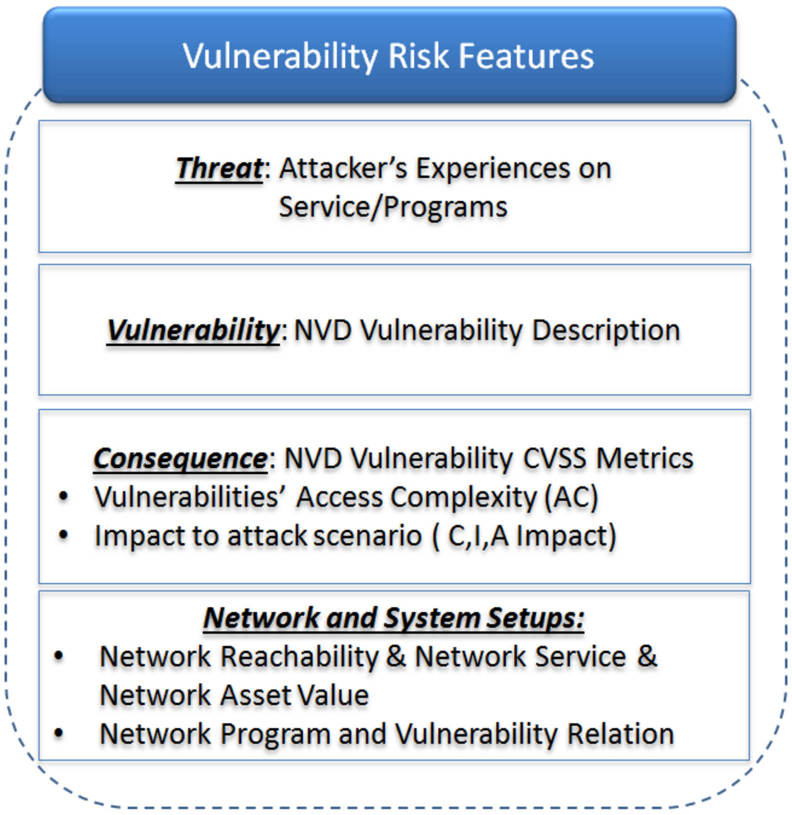
In this chapter, an overview of the proposed risk-based vulnerability prioritization framework is presented. This framework assesses a vulnerability risk based on the characterized vulnerability risk features (discussed in Chapter 4). This framework contains two stages, including the vulnerability risk assessment and the vulnerability prioritization, which are presented in Chapter 5 and 6.

Chapter 4

VULNERABILITY RISK FEATURES

This chapter illustrates the characterized vulnerability risk features in the proposed risk-based vulnerability prioritization framework.

4.1 Characterized Vulnerability Risk Features Overview



Vulnerability Risk Features
Threat: Attacker's Experiences on Service/Programs
Vulnerability: NVD Vulnerability Description
Consequence: NVD Vulnerability CVSS Metrics
• Vulnerabilities' Access Complexity (AC)
• Impact to attack scenario (C,I,A Impact)
Network and System Setups:
• Network Reachability & Network Service & Network Asset Value
• Network Program and Vulnerability Relation

Figure 4.1: The Characterized Vulnerability Risk Features

As shown in Figure 4.1, vulnerability risk features are characterized by considering threat, vulnerability, consequence, and network and system setups. The vulnerability risk is modeled by several vulnerability risk features that are based on the *Threat, Vulnerability, and Consequence* risk framework (Kaplan and Garrick, 1981). The threat refers to a harmful event and is collected from both the historical attack reports in a computer networking system and the vulnerability-exploit records in the wild (e.g., ExploitDB (The Offensive Security, 2020)). The vulnerability is identified by NVD (The National Institute of Standards and Technology, 2020), the consequence is from CVSS metrics (The Forum of Incident Response and Security Teams, 2021), and the network and system setups are from a computer networking system, where vulnerabilities will be patched sequentially based on the prioritization ranks from high to low. As shown in Figure 4.1, a threat profile includes threat, vulnerability, consequence, and network and system setups. The Cybersecurity and Infrastructure Security Agency (CISA) highlights that a threat profile, that includes "characterization of likely intent, capability, and target of threats to the function," can be used to guide the risk analysis (The Cybersecurity and Infrastructure Security Agency, 2016). Under this motivation, the previous study attempts to use the threat profile to customize risk prioritization by tuning vulnerability risk assessment to a known attacker group. The threat is defined as the most relevant information determined from current and historical experience on the defended network, because "attackers may prefer to reuse an existing exploit and make necessary changes over developing a new exploit from scratch" (Bao *et al.*, 2017), tuning a threat based on different sets of threat profiles might lead to a more accurate prediction of vulnerability risk prioritization (Alperin *et al.*, 2019).

4.2 Relations to Risk-based Vulnerability Prioritization Framework

The characterized vulnerability risk features contribute to the proposed risk-based vulnerability prioritization framework as shown in Figure 4.2, where the vulnerability

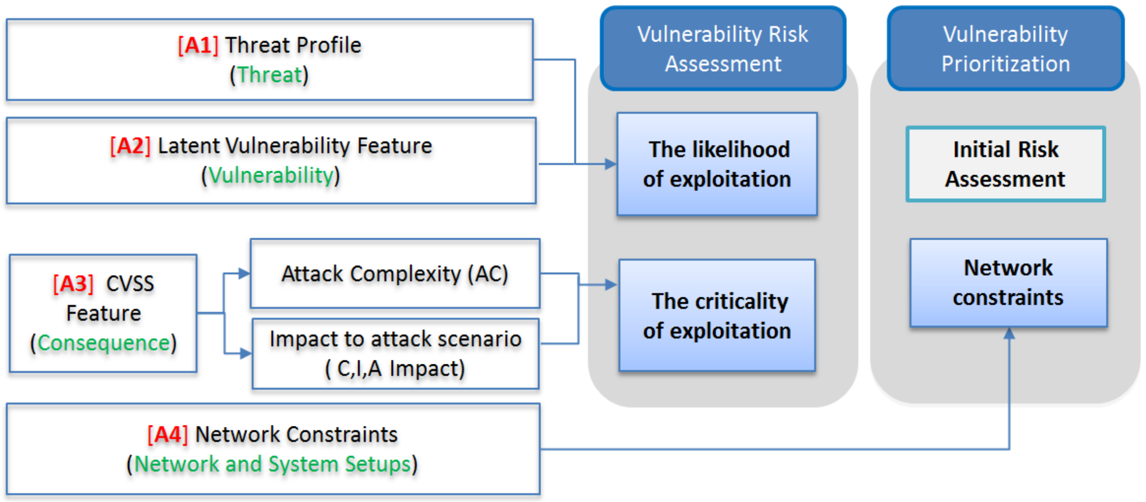

[A1] Threat Profile
(Threat)
[A2] Latent Vulnerability Feature
(Vulnerability)
[A3] CVSS Feature
(Consequence)
Attack Complexity (AC)
Impact to attack scenario
(C,I,A Impact)
[A4] Network Constraints
(Network and System Setups)
Vulnerability Risk Assessment
The likelihood of exploitation
The criticality of exploitation
Vulnerability Prioritization
Initial Risk Assessment
Network constraints

Figure 4.2: The Relations of the Characterized Vulnerability Risk Features to the Proposed Framework

risk features are learned and analyzed by the proposed vulnerability risk assessment model and the proposed vulnerability prioritization reasoning engine in this framework. Four arguments are proposed to construct the relations of the characterized vulnerability risk features to this proposed framework.

- *Argument 1* **(A1)** the threat profile (derived from both the threat of a defended network and the exploits record in the wild) has attributes on identifying attackers' experiences of software services, which contributes to measure the likelihood of exploitation;

- *Argument 2* **(A2)** the latent vulnerability feature (derived from the vulnerability description) contributes to measure the likelihood of exploitation;

- *Argument 3* **(A3)** the CVSS feature (derived from the threat of a defended network) has attributes on identifying access complexity (AC) and three impact metrics (confidentiality impact, integrity impact, and availability impact (C, I, A)), which contributes to measure the criticality of exploitation; and

- *Argument 4* **(A4)** the network profile (derived from network and system setups) can affect a vulnerability's security impact in a computer networking system, which constrains the vulnerability risk assessment.

These proposed arguments are validated as follows. For argument 1 (**A1**), attackers prefer to maximize their gains by leveraging costs and gains when using vulnerabilities in attacks (Allodi *et al.*, 2017). The more difficult it is to exploit vulnerabilities, the higher the costs attackers pay; the more significant losses to victims when attacked, the higher the gains attackers obtain. Thus, an attacker's knowledge of the experienced services/systems (e.g., the software set in the threat modeling method in Section 5.1.2) may reduce associated costs on learning services and result in increasing the likelihood of exploitation. Furthermore, the previous study (Alperin *et al.*, 2019) proves that the exploits record in the wild is the critical label for training the supervised risk models to assess how likely a vulnerability is to be exploited. Thus, the exploits record has attributes on assessing the likelihood of exploitation.

For argument 2 (**A2**), the latent vulnerability feature is derived from the vulnerability textual description by utilizing the natural language processing technique of latent semantic analysis. Such vulnerability textual description indicates the vulnerable components in product and versions, the attacker's type, the attack vector, etc. The previous study successfully utilizes the latent vulnerability feature to measure the likelihood of exploitation for vulnerability prioritization (Alperin *et al.*, 2019).

For argument 3 (**A3**), the CVSS vector of AC, C, I, A works as an indicator for analyzing the exploits (Allodi and Massacci, 2017; Ross *et al.*, 2017). The AC indicates the costs of an 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., how severe/critical impact is on the defended network when exploiting the vulnerability), and then contributes to measure the criticality of exploitation in the network.

For argument 4 (**A4**), the security impact of a vulnerability on a network is closely associated with interactions among vulnerabilities and network environment elements (Chung *et al.*, 2013; Lai and Hsia, 2007; Ou *et al.*, 2005). A vulnerability's security impact in a computer networking system can be affected by the status of a service associated with this vulnerability directly. For example, for service A in a machine that is totally isolated in the network, this means that in this machine, this vulnerability cannot actually be exploited by attackers due to the lack of access paths. In this situation, the security impact of this vulnerability might be constrained by this specific network and system setup. Therefore, the network profile derived from network and system setups can affect the vulnerability risk assessment.

4.3 Summary

In this chapter, the characterized vulnerability risk features are presented. This chapter also explains how these vulnerability risk features contribute to the proposed risk-based vulnerability prioritization framework. Based on the findings in the existing studies, the proposed arguments for vulnerability risk features are validated.

Chapter 5

VULNERABILITY RISK ASSESSMENT

In this chapter, a new vulnerability risk assessment model is proposed, called LICALITY. 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"(Allodi and Massacci, 2014). The existing AI-based approaches assess risk by estimating how vulnerabilities' features are associated with the exploits in the wild (Alperin *et al.*, 2019; Sabottke *et al.*, 2015; Allodi and Massacci, 2014; Jacobs *et al.*, 2020), which have limitations that the estimation of risk varies on different datasets and ML/DL models (Bullough *et al.*, 2017). To overcome these limitations, LICALITY is proposed to assess the risk from the "LIkelihood and critiCALITY" of exploitation. LICALITY improves the existing vulnerability risk assessment solutions by developing a novel threat modeling method based on three arguments (A1,A2,A3) discussed in Chapter 4. LICALITY captures and learns the threat attributes from a given historical threat in a computer networking system.

5.1 Proposed Vulnerability Risk Assessment Model

LICALITY assesses a vulnerability's risk by consolidating *the criticality of exploitation* derived from its CVSS features with *the likelihood of exploitation* derived from a vulnerability's latent semantic analysis (LSA) features. LICALITY utilizes a neuro-symbolic technique with neural network and probabilistic logic programming (NN-PLP) model to develop the risk assessment model. The neuro-symbolic computing is a novel AI technique that integrates neural networks with reasoning methods (i.e., logic and probability) (De Raedt *et al.*, 2019). This computation combines learning from the environment (on the neural network side) and reasoning from what has been learned (on the reasoning side) (Garcez *et al.*, 2019). By learning the threat features identified by the threat modeling method, LICALITY can generate the risk assessment for a vulnerability.

5.1.1 Background and Motivation

In this section, first, the background of neuro-symbolic computing is introduced. Secondly, a motivation example for the neuro-symbolic computing-based risk assessment model is demonstrated in Figure 5.1.

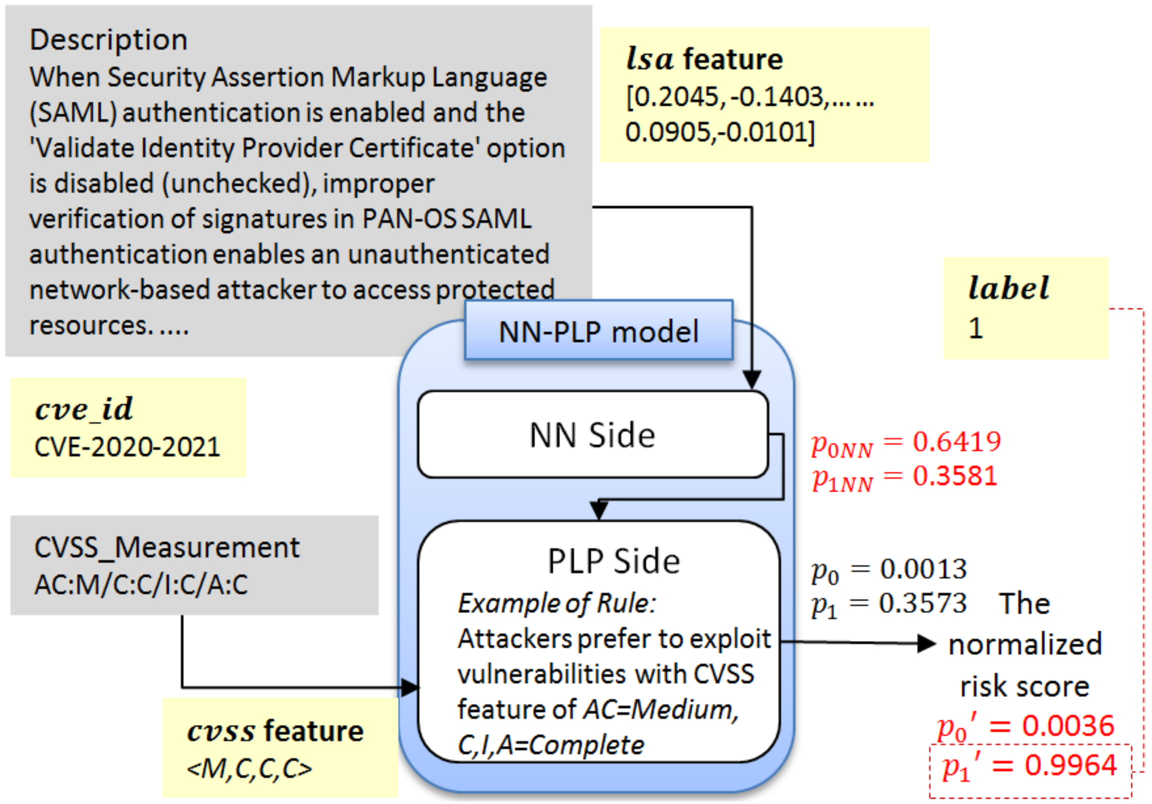
The Background of Neuro-symbolic Computing The neuro-symbolic computing model combines two fundamental cognitive abilities: learning from environment and reasoning from what has been learned (Garcez *et al.*, 2019). The neuro-symbolic model in DeepProblog (Manhaeve *et al.*, 2018) contains the neural network side and the symbolic side, where the *neural predicates* is used as interface between the symbolic side and the neural side. Both sides treat each other as a black box. For example, the symbolic side does not know the internal parameters of the neural network but can calculate the gradient of the loss related to the output of the neural network. The neural network side can use the loss related to its output calculated by the logic side to start backpropagation and then calculates the gradient for the internal parameters. A standard gradient based optimization (eg, Adam, SGD, etc) can update the internal parameters of the network. Meanwhile, the loss gradient for logic model is used to update the learned probabilistic facts or clauses in logic program (Manhaeve *et al.*, 2018). The output comes from the logic reasoner, where comparing the predicted confidentiality of queries in all possible conditions. The higher confidentiality, the more likely this prediction is to be true. The logic reasoner will output the query that has the highest confidentiality.

The **neural network side** takes $\langle features, label \rangle$ pairs directly for training. Each $label$ is viewed as an atom q_{label}. Let $\mathbf{p}$ denote the vector of all the probabilities of the probabilistic atoms; let $P(q_{label})$ denote the probability of q_{label}, and let θ denote the neural network parameters. Note that $\mathbf{p}$ consists of neural network outputs and probabilities from probabilistic rules. Thus, once the value of $\mathbf{p}$ is obtained through the forward propagation, the value of $\frac{\partial \mathbf{p}}{\partial \theta}$ is obtained through the backpropagation. The **symbolic side** compiles all logic rules into a Sentential Decision Diagram (SDD), a tree structure where each node represents an atom. The SDD defines the logical and probabilistic relationships among atoms, and can represent a propositional knowledge base for probabilistic reasoning. Due to the logic computation only uses differentiable operators (i.e., addition, negation, and multiplication), the value of $\frac{\partial P(q_{label})}{\partial \mathbf{p}}$ can also be obtained (Manhaeve *et al.*, 2018). The chain rule in Equation 5.1 is used to back propagate the loss of neural network to its parameters θ:

$$\frac{\partial P(q_{label})}{\partial \theta} = \frac{\partial P(q_{label})}{\partial \mathbf{p}} \times \frac{\partial \mathbf{p}}{\partial \theta}. \tag{5.1}$$

The Motivation Example Figure 5.1 shows a motivation example of using LICALITY to assess CVE-2020-2021. The LSA feature is learned by an neural network (NN) model to output the likelihood of exploitation $p_{1_{NN}}$. However, 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. The vulnerability CVE-2020-2021 was reported in an advanced persistent threat (APT) attack by the Cybersecurity and Infrastructure Security Agency (CISA) (The Cybersecurity & Infrastructure Security Agency (CISA), 2020). Thus, the desired output should identify this vulnerability as risky.

To overcome these limitations, LICALITY develops the probabilistic logic pro-



Description
When Security Assertion Markup Language (SAML) authentication is enabled and the 'Validate Identity Provider Certificate' option is disabled (unchecked), improper verification of signatures in PAN-OS SAML authentication enables an unauthenticated network-based attacker to access protected resources. ....
lsa feature
[0.2045, -0.1403,…… 0.0905,-0.0101]
label
1
NN-PLP model
NN Side
$p_{0NN} = 0.6419$
$p_{1NN} = 0.3581$
cve_id
CVE-2020-2021
CVSS_Measurement
AC:M/C:C/I:C/A:C
PLP Side
Example of Rule:
Attackers prefer to exploit vulnerabilities with CVSS feature of AC=Medium, C,I,A=Complete
$p_0 = 0.0013$
$p_1 = 0.3573$
The normalized risk score
$p_0' = 0.0036$
$p_1' = 0.9964$
cvss feature
<M,C,C,C>

Figure 5.1: A Running Example for Assessing CVE-2020-2021

gramming (PLP) model to efficiently learn threat attributes identified by threat modeling. In this motivation example, the given threat is assumed that attackers prefer to exploit *Medium* complexity (AC=Medium), *High* impact vulnerabilities (C,I,A=Complete). Thus, LICALITY learns a higher probabilistic value in the PLP model for the CVSS feature of $\langle M, C, C, C \rangle$. When assessing the risk of vulnerability CVE-2020-2021, 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 5.2 discussed in Section 5.1.2). Finally, LICALITY generates the normalized risk score, where $p_1' = 0.9964 > p_0' = 0.0036$[1], which matches the desired

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

output (*label* is 1). LICALITY's output more accurately represents this vulnerability's real risk, where is summarized in Table 5.1.

Table 5.1: Comparing Outputs from NN and PLP for CVE-2020-2021

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

5.1.2 System and Model Design

This section presents the proposed study system, models, and assumptions of LICALITY, the proposed vulnerability risk assessment model. LICALITY assesses a vulnerability risk based on the *Threat, Vulnerability, and Consequence* risk framework (Kaplan and Garrick, 1981). The CVSS feature of a vulnerability is defined by the CVSS metrics and the LSA feature is derived from the vulnerability description. A novel neuro-symbolic computing-based vulnerability risk assessment model is developed to learn the CVSS feature and LSA feature. LICALITY assesses a vulnerability's risk from two perspectives as (1) *the 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) *the criticality of exploitation*: the access complexity and the impacts of successful exploitation based on CVSS metrics.

Figure 5.2 shows the overview of LICALITY's system architecture. LICALITY contains two components: neural network-probabilistic logic programming (NN-PLP)-based Learning and vulnerability risk assessment. In the NN-PLP-based learning, LICALITY processes the NVD vulnerabilities to obtain the LSA feature and

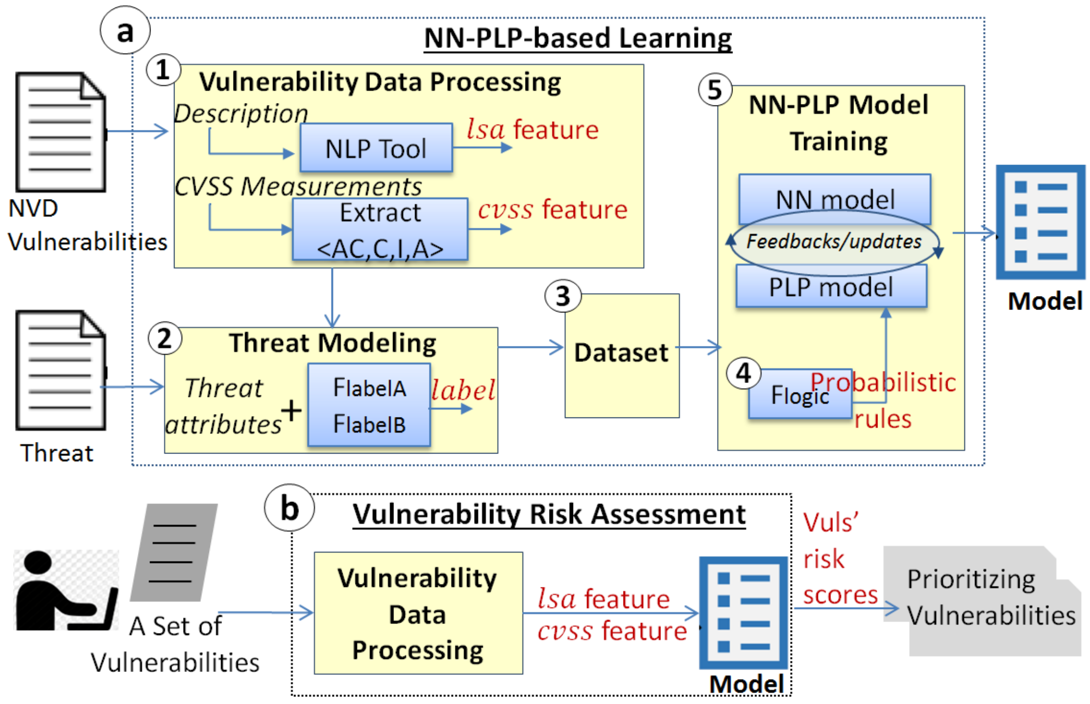


Figure 5.2: The Overview of LICALITY System Architecture

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 [2] and the labeling functions F_{labelA} and F_{labelB}. The processed LSA feature (shown as *lsa* feature in Figure 5.2), CVSS feature (shown as *cvss* feature in Figure 5.2), and label of vulnerabilities are used to construct a dataset (③). Additionally, LICALITY develops the probabilistic rules in the PLP model by the rule 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.

In the vulnerability risk assessment, a security admin can evaluate a set of vulner-

[2]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 (Mateski *et al.*, 2012).

abilities' 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 trained assessment model, LICALITY assesses the likelihood of exploitation from *lsa* features, and assesses the criticality of exploitation from *cvss* features. LICALITY consolidates the likelihood and criticality measurements to derive vulnerabilities' risk scores for the given vulnerabilities from a cyber admin.

Vulnerability Data Processing

As shown in step ① in Figure 5.2, 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 5.1. 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 NLP tool of LSA (Dumais, 2004) 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 (Alperin *et al.*, 2019). 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) (Dumais, 2004) assigns each identified term in the corpus a weight from 0 to 1.

3. Use truncated Singular Value Decomposition (SVD) algorithm (Dumais, 2004) 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 a description as a whole. TF-IDF weights 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 (Dumais, 2004). 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.

Threat Modeling

In Chapter 2, the existing risk prioritization approaches have limitations that the high CVSS base score does not realistically represent the actual exploits (Allodi and Massacci, 2014; Jacobs *et al.*, 2020); and for a specific network's defense, the record of exploits in the wild (Alperin *et al.*, 2019; Allodi and Massacci, 2014) does not associate to vulnerabilities that are more critical to the defended network (Alperin *et al.*, 2019). According to the existing study(Allodi and Massacci, 2014; Jacobs *et al.*, 2020), the most relevant threat information should be considered in this risk model. Thus, the threat modeling method is proposed based on the three arguments (A1,A2,A3). The threat modeling (shown in step ② in Figure 5.2) 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 5.1.2 shows how its $cvss$ and lsa features are obtained. The $label$ is defined according to Definition 1 to construct a dataset (shown in step ③ in Figure 5.2).

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 (The Offensive Security, 2020))*

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.*

□

A dataset $\mathcal{D}$ is constructed in step ③ in Figure 5.2, 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.

Assessment Model Training

The NN-PLP model is developed on a neuro-symbolic computation platform DeepProbLog (Manhaeve *et al.*, 2018). As shown in Figure 5.1, 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)(Garcez *et al.*, 2019). During the training processes, the label is used to compute the loss to update the NN-PLP model's parameters.

The NN side is a neural network model with a gradient-based algorithm, such as, Adam (Kingma and Ba, 2014), SGD (Wiki, 2020), 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 (Alperin *et al.*, 2019). LICALITY develops the PLP model in probabilistic logic programming (PLP) language (Manhaeve *et al.*, 2018; De Raedt *et al.*, 2007). 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**

  $nn(model\_net, lsa, l) :: l\_e(lsa, l)$

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 5.2) 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.* □

During the NN-PLP-based learning (⑧), LICALITY trains the NN-PLP model by following the parameter learning of DeepProblog (Manhaeve *et al.*, 2018). 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 (Werbos, 1990) 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 (Manhaeve *et al.*, 2018) 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 (Manhaeve *et al.*, 2018):

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

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*). These gradients for parameter learning are computed in the Sentential Decision Diagram (SDD) (Darwiche, 2011). 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 5.1, 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 (5.2) (Manhaeve *et al.*, 2018) 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.

$$\begin{aligned}&(p_{_{NN}}, \overrightarrow{v_{p_{NN}}}) \bigotimes (p_{_{PLP}}, \overrightarrow{v_{p_{PLP}}}) = \\ &(p_{_{NN}} p_{_{PLP}}, p_{_{PLP}} \overrightarrow{v_{p_{NN}}} + p_{_{NN}} \overrightarrow{v_{p_{PLP}}})\end{aligned} \tag{5.2}$$

$p_{_{NN}}$ 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 5.3 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 (ⓑ), 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 (5.2). For example, in Figure 5.3, 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, the LICALITY's normalized risk score is 0.9964, where it is calculated by the following formula $0.3573/(0.3573 + 0.0013) = 0.9964$.

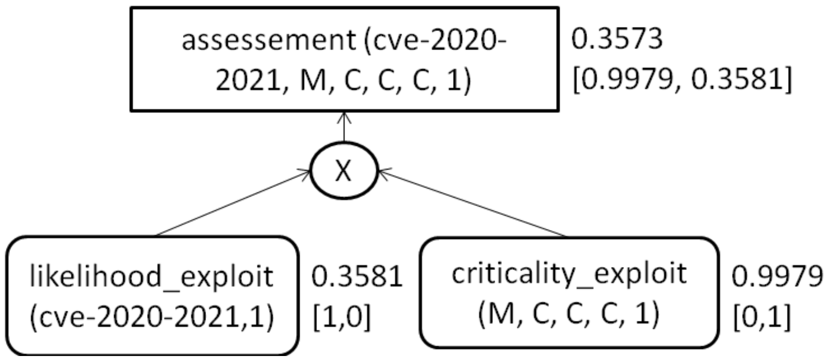

assessement (cve-2020-2021, M, C, C, C, 1)
0.3573
[0.9979, 0.3581]
X
likelihood_exploit (cve-2020-2021,1)
0.3581
[1,0]
criticality_exploit (M, C, C, C, 1)
0.9979
[0,1]

Figure 5.3: The Example of Computing in SDD in LICALITY

5.2 Experimental Evaluation

This section presents a comprehensive case study to investigate LICALITY's performance (marked as the *NN-PLP & Threat Modeling*) on risk prioritization with the existing approaches in Figure 5.9, Figure 5.10, Table 5.6, Table 5.7 and 5.8. The *Existing Expert-based Solution* represents the predominant expert-based approach that leverages the CVSS scores(Mell *et al.*, 2007) 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 (Alperin *et al.*, 2019).

Additionally, two AI techniques are investigated to prioritize vulnerabilities for remediation, including a neuro-symbolic model (the NN-PLP model) and a neural network model (the NN-only model). 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[3] as:

- Case 1 (Microsoft Vulnerabilities (MVs)): the historical threat is from the high-risk Microsoft Vulnerabilities (called MVs_2015) in the 2015 CISA alert (The Cybersecurity and Infrastructure Security Agency, 2015), and the future threat (called MVs_2020) is from the top routinely exploited Microsoft Vulnerabilities in the 2020 CISA alert (The Cybersecurity and Infrastructure Security Agency, 2020). Microsoft became cyber attackers' preferred platform. The reported Microsoft vulnerabilities have risen 64% from 2015 to 2019 (Microsoft, 2020). In Case 1, the historical threat and the future threat are associated with Microsoft products and share some of the same features (as shown in Table 5.2 and Table 5.5).

[3]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 (Allodi and Massacci, 2014; Jacobs *et al.*, 2020)

- Case 2 (Advanced Persistent Threat Vulnerabilities (APTVs)): the historical threat (called APTVs_2017) is from the APT28 attacker in the FireEye 2017 report (The FireEye, 2017), and the future threat (called APTVs_2020) is from the APT attack identified recently in the 2020 CISA alert (The Cybersecurity & Infrastructure Security Agency (CISA), 2020). The APT attack is a stealthy threat that uses continuous and sophisticated hacking techniques to access a system (Alshamrani *et al.*, 2019). Additionally, the APT attackers usually pursue their targets over months or years. Thus the APT attack is challenging to detect (Alshamrani *et al.*, 2019). 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 5.2 and Table 5.5).

5.2.1 Experimental Settings

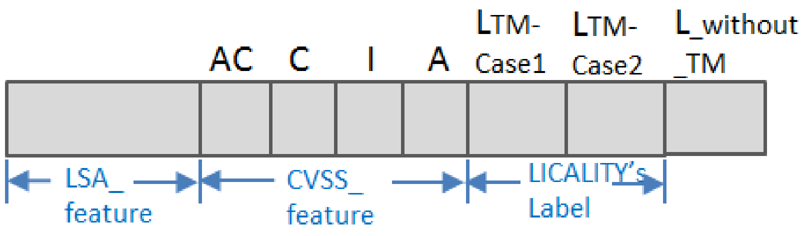


Figure 5.4: The Data Structure of the Dataset in the Evaluation

Figure 5.4 shows an overview of the dataset structure in the evaluation. This dataset contains vulnerabilities from 1999 to June 2021 in NVD (The National Institute of Standards and Technology, 2020). 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 empty CVSS records, the amount of NVD vulnerabilities are refined to total 155,176 in this chapter. A JSON parser is developed in Python to extract vulnerabilities’ CVE_IDs, descriptions, and CVSS vectors from NVD JSON files (The National Institute of Standards and Technology, 2020). The CVSS features (e.g., $X_{i+1}, ..X_{i+4}$) and the LSA features (e.g., $X_0, .., X_i$) are generated by using the vulnerability data processing function (defined in Section 5.1.2), 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 lower-casing 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 into corpus by using the frequency-inverse document frequency (TF-IDF) matrix (Dumais, 2004).

3. Transform the TF-IDF matrix into LSA features through a truncated SVD algorithm (Dumais, 2004). Different features are explored to generate the LSA features and select the best SVD explained variance ratio. In this chapter, the features number is 150, and the SVD explained variance ratio is 0.87. Such LSA features are shown as $(X_0, ..., X_i), i = 149$ in the dataset.

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

- $L\_TM - Case1$: labels the risk 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 the risk 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 the risk of vulnerabilities with the given exploits from ExploitDB without threat modeling.

For $L\_TM - Case1$ and $L\_TM - Case2$, the dataset is labeled by using the labeling function F_{labelA} (defined in Section 5.1.2) with the service set and the CVSS set of the historical threat. Then, labels are generated 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 (The Cybersecurity and Infrastructure Security Agency, 2015), which advises IT and security professionals to prioritize patching these most commonly known vulnerabilities (The Cybersecurity and Infrastructure Security Agency, 2016). 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 (The FireEye, 2017). 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 $L\_without\_TM$, the ground truth are the exploits records in ExploitDB(The Offensive Security, 2020) as the existing study (Edkrantz *et al.*, 2015; Alperin *et al.*, 2019). A vulnerability's $L\_without\_TM$ is 1 if there is a corresponding exploit record in the ExploitDB; otherwise, it is 0.

Table 5.2 shows the details of threats MVs_2015 and APTVs_2017, where the risky vulnerabilities ($Risky\_vul$), software set (S_{sw}), and CVSS set (S_{cvss}) are listed.

Table 5.2: The Historical Threat of Case 1 and Case 2

The historical threat	Risky vulnerabilities ($Risky\_vul$)	The software set (S_{sw})	The CVSS set* (S_{cvss})
MVs_2015 (The Cybersecurity and Infrastructure Security Agency, 2015) ($Case1$)	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 (The FireEye, 2017) ($Case2$)	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.

Table 5.3: An Overview of the Dataset

Property	Dataset
Vulnerability	NVD (The National Institute of Standards and Technology, 2020)
Exploits in the wild	ExploitDB (The Offensive Security, 2020)
Historical threat	MVs_2015 (The Cybersecurity and Infrastructure Security Agency, 2015) (Case 1), APTVs_2017 (The FireEye, 2017) (Case 2)
Future threat	MVs_2020 (The Cybersecurity and Infrastructure Security Agency, 2020) (Case 1), APTVs_2020 (The Cybersecurity & Infrastructure Security Agency (CISA), 2020) (Case 2)
Positive labels	38,036 ($L\_TM - Case1$), 65,871 ($L\_TM - Case2$) , 11,679 ($L\_without\_TM$)

Table 5.3 shows that there are 11,679 positive labels with $L\_without\_TM$, 38,036 positive labels with $L\_TM - Case1$, and 65,871 positive labels with $L\_TM - Case2$. In this chapter, 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 points at the rate of 80%, 10%, and 10%, respectively.

Figure 5.5 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. Figure 5.6 shows the distribution of the popular CVSS features in the datasets, where all listed CVSS features in this figure are associated with over 100 vulnerabilities among all 155,576 vulnerabilities in the dataset. Figure 5.6 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 with over 10,000 vulnerabilities in the dataset. In LICALITY, the PLP side of the NN-PLP model is trained with the CVSS features.

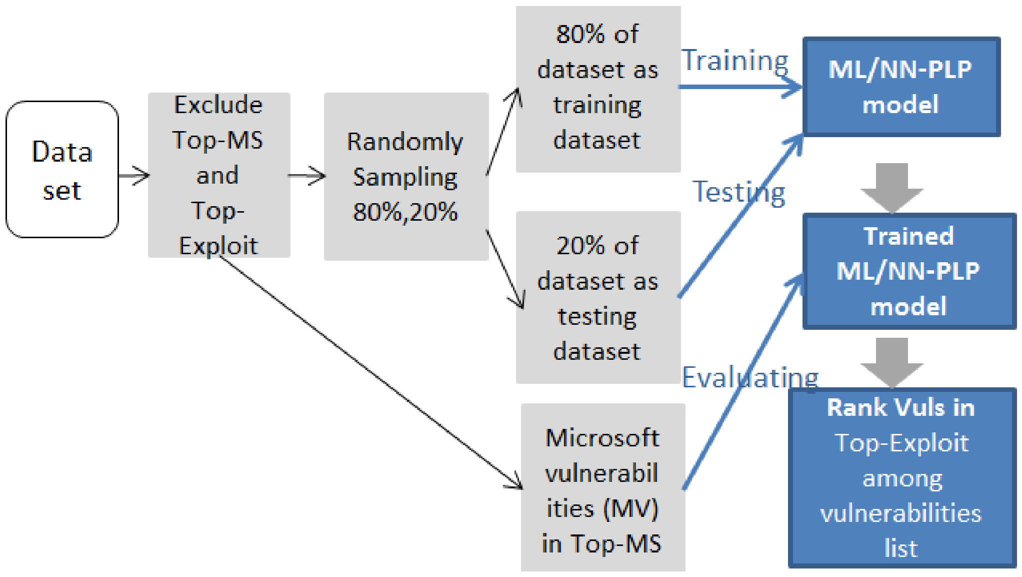

Data set
Exclude Top-MS and Top-Exploit
Randomly Sampling 80%,20%
80% of dataset as training dataset
20% of dataset as testing dataset
Microsoft vulnerabilities (MV) in Top-MS
Training
Testing
Evaluating
ML/NN-PLP model
Trained ML/NN-PLP model
Rank Vuls in Top-Exploit among vulnerabilities list

Figure 5.5: An Overview of Separating the Dataset

Performance Measurements

Risk prioritization forms a basis for allocating resources, where the high-ranked vulnerabilities will be prioritized for taking defensive actions. The performance evaluation 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 (Alperin *et al.*, 2019) uses this metric to evaluate the efficiency of machine learning algorithms as a classifier for a dataset in risk prioritization. In this section, the efficiency of using the NN-PLP model or NN-only model as a classifier for the dataset is evaluated by AUC.

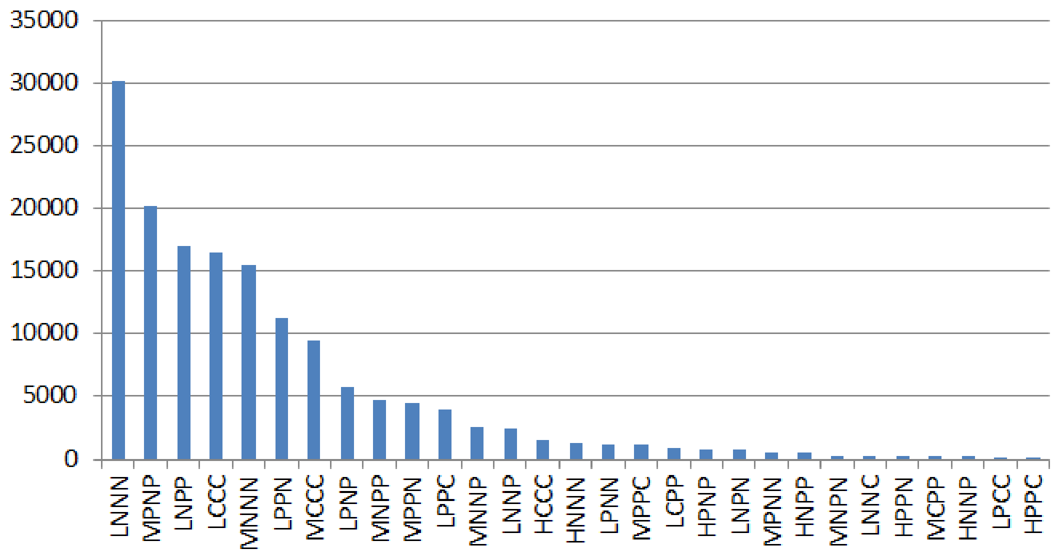


Figure 5.6: An Overview of CVSS Features' Distribution in the Dataset

2. The recommend rankings (percentage) of the assessed vulnerability among all vulnerabilities in the dataset: the existing study (Alperin *et al.*, 2019) uses this measurement to evaluate the performance of risk prioritization solutions. The recommend rankings (percentage) of the future threats (e.g., MVs_2020 in Case 1 and APTVs_2020 in Case 2) are compared among all vulnerabilities in the dataset under LICALITY (marked as *Threat Modeling*) and the existing approaches (marked as *ExploitDB* and *CVSS Score*) in Table 5.7 and 5.8.

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).

Model Implementation

This section demonstrates the implementation of models in this chapter. There are two types of models as follows:

- *NN-only model*: A neural network model with a gradient-based algorithm. The neural network model is developed 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, the NN-PLP model's NN side (e.g., neural network structure and the initial parameter settings) is 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 5.7 shows a developed four-layer neural network model, which has 150 nodes in the input layer, 100 nodes in the first hidden layer, and 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 (Alperin *et al.*, 2019), 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. The vulnerability

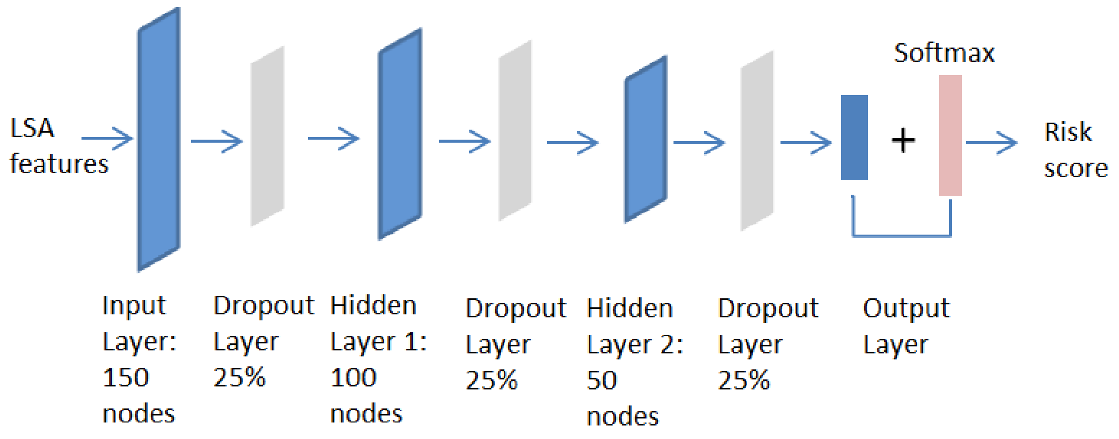


Figure 5.7: The NN-only Model in the Evaluation

CVE-2020-2021 in Figure 5.3 is illustrated as an example to explain 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 5.1.2 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:

$assessment(cve-2020-2021, M, C, C, C, 0) \leftarrow$

$l\_e(lsa, 0)$, $c\_e(M, C, C, C, 0)$;

$assessment(cve-2020-2021, M, C, C, C, 1) \leftarrow$

$l\_e(lsa, 1)$, $c\_e(M, C, C, C, 1)$,

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 $assessment(cve-2020-2021, M, C, C, C, 1)$.

5.2.2 Results

In the comparative study, the efficiency of using the neural network model (NN-only model) is compared 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. The best working scenario of LICALITY is investigated by assessing the future threats in two cases.

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 (Pytorch, 2021) 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 over-fitting. The model with minimal validation loss is selected as the trained model.

The NN-only model shown in Figure 5.7 applies the Multilayer Perceptron (MLP) algorithm. In this chapter, another gradient-based algorithm (Alperin *et al.*, 2019) (logistic regression) for learning LSA features is investigated as well. 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 (Wiki, 2020). By following the previous study (Alperin *et al.*, 2019), 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 using the MLP algorithm. Additionally, the optimizer of Adam (Kingma and Ba, 2014) is investigated as well. Since the SGD performs better on the testing set, the SGD is selected as the optimizer of the NN-only model. Nonetheless, future work can explore a more complicated NN model. The AUC of MLP and logistic regression in this evaluation both are better than what was reported in the previous study (Alperin *et al.*, 2019), so the NN-only model is well trained for lsa features and is a valid comparative model for LICALITY.

Training NN-PLP model

The NN-PLP model is constructed on the platform of DeepProbLog (Manhaeve *et al.*, 2018). Thus the 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 5.6 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. The NN-PLP model is trained with a gradient-descent-based optimizer as SGD. Table 5.4 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 5.8. The high AUC indicates that the trained NN-PLP models can classify very well on the testing set verse the NN-only model for the risk identified by threat modeling.

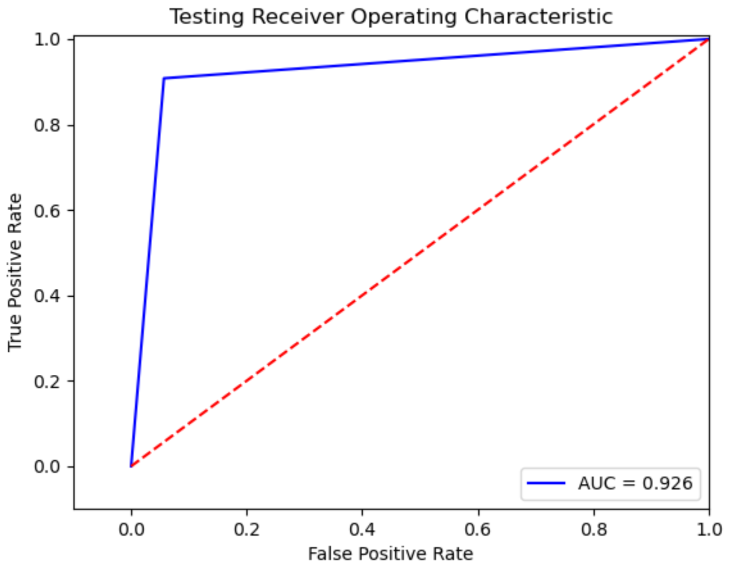


Figure 5.8: The ROC Curve of LICALITY in Case 2

Vulnerability Risk Assessment

In Case 1, the Microsoft vulnerabilities of the MVs_2020 are prioritized, where the associated threat is identified by a CISA alert in 2020 (The Cybersecurity and Infrastructure Security Agency, 2020). As shown in Table 5.5, 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, the vulnerabilities of APTVs_2020 are prioritized, where the asso-

Table 5.4: Comparing the NN-only Model and the NN-PLP Model on Classifying Risk

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

ciated threat is identified by a CISA alert in 2020 (The Cybersecurity & Infrastructure Security Agency (CISA), 2020). 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. The changes of $\langle AC, C, I, A\rangle$ vectors in CVSSv3 are compared as well, 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.

The percentages of MVs_2020 and APTVs_2020's vulnerabilities that rank among all 155,176 vulnerabilities in the dataset are compared. Viewing ranked vulnerabilities in descending order, the higher the risk score, the lower the percentage is shown in Table 5.7 and Table 5.8. A vulnerability with the lowest percentage will first be recommended for remediation. These percentages for *Existing Expert-based Solution* and *Existing ML/DL-based Solution* are computed as well. The *Existing Expert-based Solution* ranks vulnerabilities based on the CVSS score (Mell *et al.*, 2007). The *Existing ML/DL-based Solution* follows the previous study (Alperin *et al.*, 2019) that integrates the attacker model into risk prioritization.

Table 5.5: The Software Services and CVSS Vector of Vulnerabilities for Evaluation (Future Threat) in Case 1 and Case 2

Case	CVE ID	The software service	The CVSS(v2) feature [1]	The CVSS(v3) feature [1]
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.

Table 5.6: Summary of Vulnerabilities Recommended for Remediation in Case 1 and Case 2

Case	The Mitigated Vulnerabilities	Existing Expert-based Solution (Mell *et al.*, 2007)%	Existing ML/DL-based Solution (Alperin *et al.*, 2019)%	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

Figure 5.9 and Figure 5.10 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 5.9 and Figure 5.10 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 5.9 shows that LICALITY outperforms the *Existing Expert-based Solution (Mell* et al.*, 2007)* and the *NN-PLP & Without threat modeling* on correctly ranking MVs_2020 on the top position (with the lower bar height shown in Figure 5.9) for mitigation. Additionally, LICALITY has slightly better performance than the *Existing ML/DL-based Solution (Alperin* et al.*, 2019)* on classifying the risk identified by *Threat Modeling* ($L_{TM-Case1}$) in Case 1. Table 5.6 shows the details of

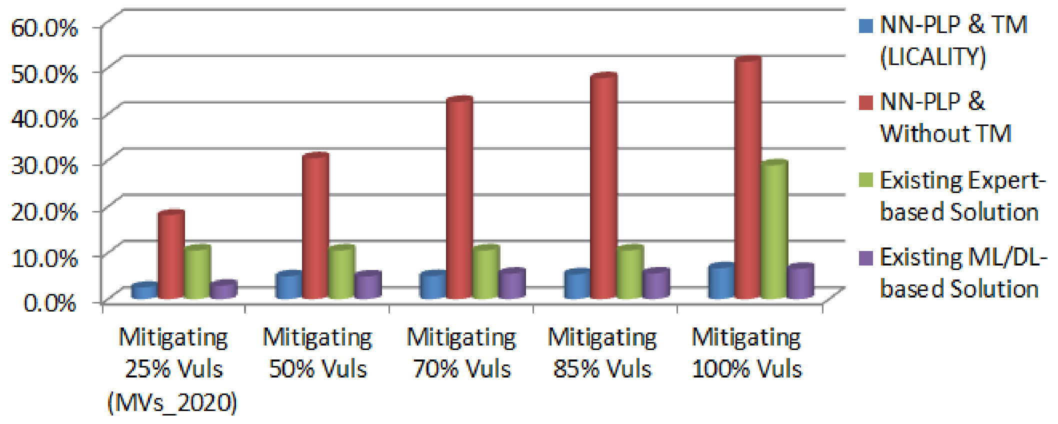


Figure 5.9: Comparing the Prioritization Ranks of MVs_2020 in Case 1

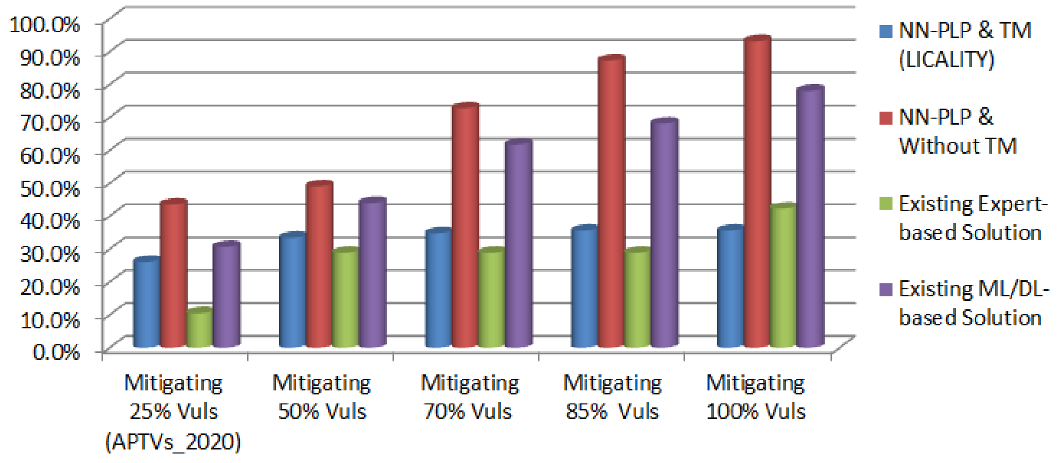


Figure 5.10: Comparing the Prioritization Ranks of APTVs_2020 in Case 2

Figure 5.9. For example, if remediating 70% of Microsoft vulnerabilities in Table 5.7 and Table 5.8, LICALITY correctly places them on the top 4.9736% of all 155,176 vulnerabilities, while the existing expert-based solution (Mell *et al.*, 2007) 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 (Mell* et al.*, 2007)* 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 (Alperin* et al.*, 2019).* In Case 2, Figure 5.10 shows that LICALITY outperforms the existing two solutions on vulnerability risk prioritization. As shown in Table 5.6, when remediating 100% of vulnerabilities of APTVs_2020, LICALITY reduces the vulnerability remediation work required by the *Existing Expert-based Solution (Mell* et al.*, 2007)* by a factor of 1.85 (over 47K vulnerabilities), and reduces such remediation work required by the *Existing ML/DL-based Solution (Alperin* et al.*, 2019)* by a factor of 2.19 (over 65K vulnerabilities, i.e., calculated as $155176 * (66.1836\% - 35.6226\%) = 47423; 155176 * (78.1082\% - 35.6226\%) = 65927$).

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 (Alperin *et al.*, 2019) is the first study to encode an attacker model's (APT28) target software service into risk prioritization through labeling. In the evaluation, LICAILITY is compared with the *existing ML/DL-based model (Alperin* et al.*, 2019)* in Section 5.2). LICALITY can almost reach the 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 evaluation, LICALITY 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 the historical threat APTVs_2017, so the historical threat's attributes on the attacker's strength and preference (encoded as the software set (S_{sw})) might make it 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 (The Cybersecurity & Infrastructure Security Agency (CISA), 2020). 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 (Mell* et al.*, 2007)* by a factor of 1.85, and reduces such remediation work required by the *Existing ML/DL-based Solution (Alperin* et al.*, 2019)* by a factor of 2.19.

Details of the risk scoring and ranking data are shown in Tables 5.7 and 5.8, where the risk scores and the associated rankings are the updated assessments by updating the $\langle C, I, A\rangle$ to CVSSv3.

5.3 Summary

In this chapter, a novel vulnerability risk assessment model is proposed. This vulnerability risk assessment model assesses a 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. The best working scenario of LICALITY is investigated by comparing the corresponding risk ranking results in two cases. LICALITY (NN-PLP model & threat modeling method) successfully prioritizes vulnerabilities (refers to the future threat) on the top positions in the evaluation.

Table 5.7: Comparing Risk Scores and Ranking (%) of MVs_2020 and APTVs_2020 in Case 1

Case	CVE ID	Existing Expert-based Solution (Mell *et al.*, 2007)	Existing ML/DL-based Solution (Alperin *et al.*, 2019)	NN-PLP & Without Threat Modeling	NN-PLP & Threat Modeling (LICALITY)	Existing Expert-based Solution (Mell *et al.*, 2007)%	Existing ML/DL-based Solution (Alperin *et al.*, 2019)%	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

Table 5.8: Comparing Risk Scores and Ranking (%) of MVs_2020 and APTVs_2020 in Case 2

Case	CVE ID	Existing Expert-based Solution (Mell *et al.*, 2007)	Existing ML/DL-based Solution (Alperin *et al.*, 2019)	NN-PLP & Without Threat Modeling	NN-PLP & Threat Modeling (LICALITY)	Existing Expert-based Solution (Mell *et al.*, 2007)%	Existing ML/DL-based Solution (Alperin *et al.*, 2019)%	NN-PLP & Without Threat Modeling %	NN-PLP & Threat Modeling (LICALITY)%
Case	CVE-2018-13379	5	0.0891	0.0124*	0.9981*	66.1836	78.1082	36.8017	34.7993
2	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 Table5.5.

Chapter 6

VULNERABILITY PRIORITIZATION

In this Chapter, a prioritization reasoning engine ILLATION is proposed. The security impact of a vulnerability on a network is closely associated with interactions among vulnerabilities and network environment elements (Ou *et al.*, 2005; Lai and Hsia, 2007; Chung *et al.*, 2013). However, the existing network agnostic approaches (i.e., the expert-based approaches and the AI-based approaches) either use predefined measurements and static metrics (Mell *et al.*, 2006, 2007), or do not consider actual network setup and constraints (Spring *et al.*, 2018; Shah *et al.*, 2019; Chen *et al.*, 2019; Sabottke *et al.*, 2015; Alperin *et al.*, 2019; Zeng *et al.*, 2021; Bozorgi *et al.*, 2010). The network specific approaches (i.e., the attack graph-based approaches (Tupper and Zincir-Heywood, 2008; McQueen *et al.*, 2006; Lai and Hsia, 2007; Jajodia *et al.*, 2005)) usually have high complexity, which hinder security admins in adopting these approaches for vulnerability prioritization. The vulnerability's features (i.e., the likelihood of vulnerability exploitation, attacker's preference, etc.) cannot been covered in attack graphs as well. By considering the effects of particular network environmental elements (e.g., running service, node reachability, asset value, etc.) on a vulnerability's security impact, ILLATION enhances the existing network agnostic approaches. ILLATION ranks out vulnerabilities that have security influences on the critical assets by refining the initial assessment with two network constraints. Such constraints are derived from network profiles of a network and security setup (i.e., network reconfiguration, access control, firewall rules updates, etc.).

6.1 Proposed Vulnerability Prioritization Reasoning Engine

Prioritizing vulnerability under different network profiles is needed in an organization. This is due to cyber defenders usually adjusting the network setups (e.g., changing firewall rules, suspending services, etc.) to accommodate various reasons for postponing vulnerability remediation without increasing the entire system risk.

6.1.1 Background and Motivation

First, this section presents an illustrative example of how ILLATION prioritizes vulnerabilities before and after mitigation.

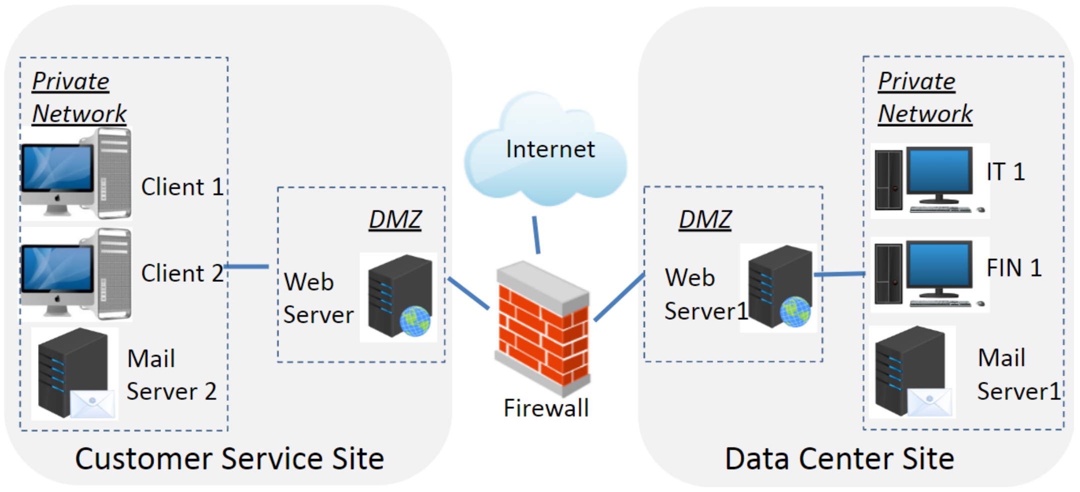


Figure 6.1: The Network Segmentation Overview in the Illustrative Example.

Figure 6.1 describes a mock enterprise network system established on a private cloud (i.e., Amazon AWS). The network system consists of two virtual networks: a Data Center Site and a Customer Service Site, which can access the services on each site through the internet. Several security policies are deployed in this network environment. In the demilitarized zone (DMZ) of this network segmentation, virtual machines (VMs) have limited access to the private network. As shown in Figure 6.1, there are 8 virtual machines in this network segmentation. Vulnerability scanning tools (e.g., Nessus (Wiki, 2021), Nmap (Lyon, 2008), etc.) detect vulnerabilities in this network. Table 6.2 shows 9 scanned vulnerabilities, where mail servers, web servers, and database servers have the same vulnerabilities. Table 6.1 presents the details of network and system setups in this illustrative example.

Table 6.1: The Example of Security Rules Configuration for NP-1 and NP-2.

Network Profile	Destination VM	Source VM	Protocol	Port	Status	Mitigated Vulnerabilities
NP-1	IT1	WebServer1, WebServer2, MailServer1	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 80,443, 993	OPEN	N/A
NP-1	FIN1	WebServer1, MailServer1	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 80, 443, 993	OPEN	N/A
NP-1	WebServer1	IT1, FIN1, MailServer1	TCP, IMAP	80,443, 1433, 3306, 5439, 5432, 1521,993	OPEN	N/A
NP-1	MailServer1	IT1, FIN1, WebServer1	TCP, IMAP	80,443, 1433, 3306, 5439, 5432, 1521,993	OPEN	N/A
NP-1	Client1	WebServer2	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 80, 443, 993	OPEN	N/A
NP-1	Client2	MailServer2	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 80, 443, 993	OPEN	N/A
NP-1	WebServer2	IT1, Client1	TCP, IMAP	993, 80, 443, 1433	OPEN	N/A
NP-1	MailServer2	Client2	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 80, 443, 993	OPEN	N/A
NP-2	WebServer1	IT1, FIN1, MailServer1	TCP, IMAP	80,443, 1433, 3306, 5439, 5432, 1521,993	OPEN	N/A
NP-2	MailServer1	IT1, FIN1, WebServer1	TCP, IMAP	80,443, 1433, 3306, 5439, 5432, 1521,993	OPEN	N/A
NP-2	Client1	WebServer2	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 80, 443, 993	OPEN	N/A
NP-2	Client2	MailServer2	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 80, 443, 993	OPEN	N/A
NP-2	WebServer2	IT1, Client1	TCP, IMAP	993, 80, 443, 1433	OPEN	N/A
NP-2	MailServer2	Client2	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 80, 443, 993	OPEN	N/A
NP-2	IT1	MailServer1	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 993	OPEN	CVE-2000-1221, CVE-2008-4306
NP-2	FIN1	MailServer1	TCP, IMAP	1433, 3306, 5439, 5432, 1521, 993	OPEN	CVE-2018-8626
NP-2	IT1	WebServer1, WebServer2	TCP, IMAP	80, 443, 1433, 3306, 5439, 5432, 1521,993	CLOSE	CVE-2000-1221, CVE-2008-4306
NP-2	FIN1	WebServer1	TCP, IMAP	80, 443, 1433, 3306, 5439, 5432, 1521,993	CLOSE	CVE-2018-8626

Network Profiles under Mitigation Actions

In practice, mitigation actions may include various security defense solutions at the software level (e.g., software upgrades/patches) and/or network level (e.g., changing firewall rules). The selection and evaluation of mitigation actions can be guided by comparing the intrusiveness and cost of countermeasures (e.g., traffic redirection, deep packet inspection, IP address change, etc.) (Chung *et al.*, 2013). For example, to reduce the likelihood of a vulnerability being exploited, some mitigation actions might change firewall rules or enforce more strict access control policies. In this illustrative example, two network profiles (NP-1 and NP-2) represent the network environments before and after mitigation. Figure 6.2 shows the network reachability in the NP-2, where the dashed line represents the blocked connections compared to the NP-1.

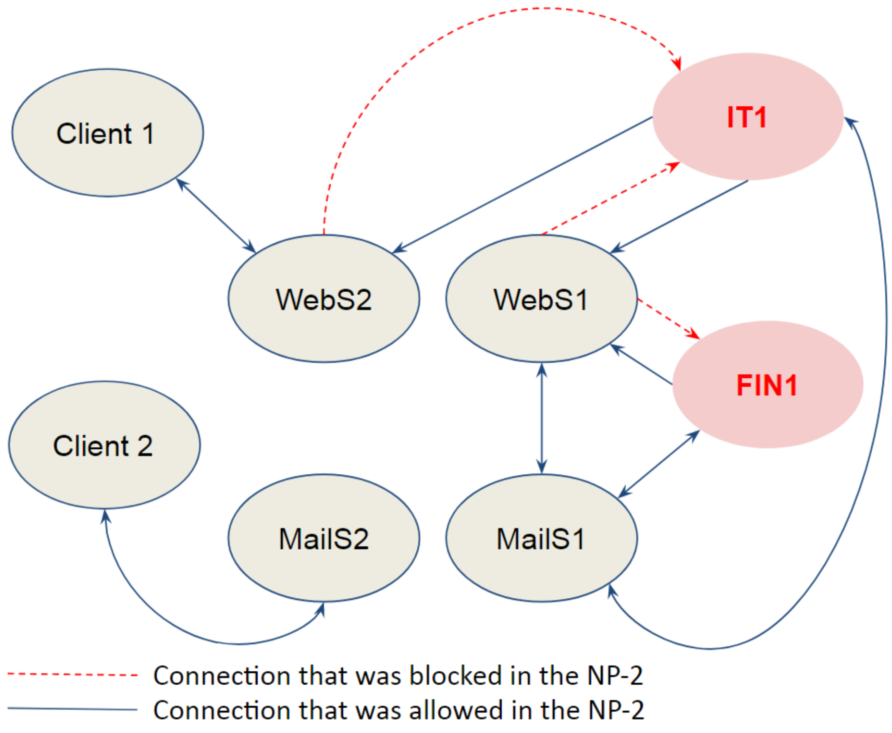

IT1
Client 1
WebS2
WebS1
FIN1
Client 2
MailS2
MailS1
Connection that was blocked in the NP-2
Connection that was allowed in the NP-2

Figure 6.2: The Network Reachability of NP-2 in the Illustrative Example.

Because ILLATION places CVE-2000-1221 (on IT1), CVE-2008-4306 (on IT1), and CVE-2018-8626 (on FIN1) in NP-1 (marked in red in Table 6.2) on the top for remediation recommendation. Cyber admins can reduce the associated risk of these vulnerabilities by reducing the attack surface of IT1 and FIN1. The common mitigation actions contain: (1) enforcing security control by specifying which types of traffic are allowed. The inbound and outbound access control rules are changed; (2) enabling security control on a specific VM that is identified as the highest risk resource. In NP-1, FIN1 is directly reachable by WebServer1 and MailServer1, and IT1 is directly reachable by WebServer1, WebServer2, and MailServr1. In this illustrative example, two web servers are viewed as the highest risk resources for IT1 and FIN1 since they are in the DMZ and connected to the internet. To minimize the impact of mitigation actions on the daily network operation, cyber admins might work on blocking the traffic from these high risk resources (web servers) first. Under this situation, all traffics that send from WebServer1 and WebServer2 to IT1 and FIN1 are blocked until the vulnerability CVE-2000-1221, CVE-2008-4306, and CVE-2018-8626 on these VMs have been patched. As shown in Figure 6.2, in NP-2, IT1 and FIN1 are only directly reachable by MailServer1, which is in the private network, where it usually has secure and encrypted connections in the network operation.

Motivation of the Proposed Prioritization Reasoning Engine

ILLATION is motivated by the observations that the risk of a vulnerability is associated with network environmental elements (Mell *et al.*, 2006, 2007; Ou *et al.*, 2005). It considers the actual network setup and the associated constraints on vulnerability risk assessment. In Table 6.2, the output of ILLATION reflects the changes of network setups from NP-1 to NP-2. After cyber admins take mitigation actions, ILLATION's output reflects the changes to the network environment as the changes to the target hosts' vulnerability risk ranks. In this illustrative example, the risk ranks of vulner-

Table 6.2: The Vulnerability Risk Ranks in the Illustrative Example

VM	**Asset Value**	**Vulnerability on this VM**	**For NP-1**[*]	**For NP-2** [*]
IT1	10	CVE-2000-1221	0.9997 (1)	0.4998 (4) (↓)
WebServer2	8	CVE-2010-3972	0.7997 (3)	0.7997 (2) (↑)
WebServer1	7	CVE-2010-3972	0.6998 (4)	0.6998 (3) (↑)
FIN1	10	CVE-2018-8626	0.9997 (1)	0.4998 (4) (↓)
Client1	8	CVE-2016-1930	0.7997 (3)	0.7997 (2) (↑)
IT1	10	CVE-2008-4306	0.9997 (1)	0.4998 (4) (↓)
MailServer2	9	CVE-2007-0671	0.8997 (2)	0.8997 (1) (↑)
MailServer1	10	CVE-2007-0671	0.4998 (5)	0.4998 (4) (↑)
Client2	8	CVE-2017-8487	0.7997 (3)	0.7997 (2) (↑)
WebServer2	8	CVE-2002-0150	0.1990 (7)	0.1990 (6) (↑)
WebServer1	7	CVE-2002-0150	0.1741 (8)	0.1741 (7) (↑)
MailServer2	9	CVE-2004-0284	0.2571 (6)	0.2571 (5) (↑)
MailServer1	10	CVE-2004-0284	0.1429 (9)	0.1429 (8) (↑)

[*] Using ILLATION, the arrow after each ranking value represents its increasing or decreasing status compared to the ranking value in the previous column.

abilities CVE-2000-1221 (on IT1), CVE-2008-4306 (on IT1) and CVE-2018-8626 (on FIN1) are affected by mitigation actions, and have been lowered down.

To prioritize vulnerabilities in this example, ILLATION initially assesses vulnerabilities in vulnerability scanning reports by using the existing risk-based vulnerability assessment model (Zeng *et al.*, 2021). And then, ILLATION embeds the network and security setups as network profiles (i.e., NP-1, NP-2) and derives the network constraints for each network profile. ILLATION consolidates initial risk assessments and network constraints of each network profile by following Rule 6.2 (discussed in Section 6.1.2). In the illustrative example, NP-1 and NP-2 represent network setups before and after mitigation. The changes in network environments from NP-1 to NP-2 lead to the changes in both defense rules and node reachability statuses in the logical reasoning engine (discussed in Section 6.1.2), which might modify the defined three logic features (discussed in Section 6.1.2). In this way, the effects of network constraints on assessing the risk of a vulnerability will be adjusted (discussed in Rule 6.3 and 6.4 in Section 6.1.2).

Design Improvements

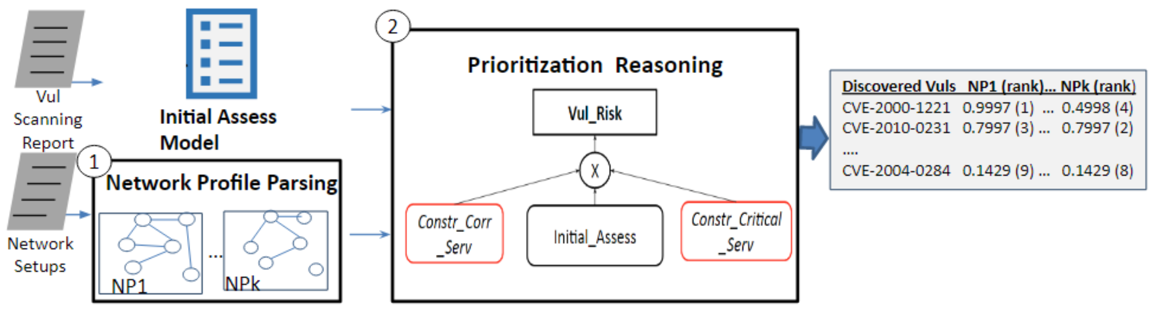
To reflect how network environment changes affect vulnerability risk assessment before and after mitigation actions in NP-1 and NP-2, ILLATION considers the network setups as an important input for the vulnerability risk assessment model, and makes several design improvements to address the design challenges discussed previously. The key improvements are briefly summarized as follows.

Key improvements: ILLATION focuses on assessing the actual effects of a vulnerability on critical assets. ILLATION ranks out vulnerabilities that have high risk and could actually affect the most critical assets by introducing two constraints of *constr_corr_serv* and *constr_critical_serv*, where the value of these constraints range from 0 to 1. The *constr_corr_serv* represents how likely a running service in a machine can be reachable by neighbor nodes in a network. For example, for service A in a machine that is totally isolated in the network (i.e., no neighbor nodes can reach this machine, no open ports for this service, etc.), the value of *constr_corr_serv* is 0. This means that in this machine, although a scanned vulnerability is able to harm this service A, it cannot actually be exploited by attackers due to the lack of access paths. Similarly, *constr_critical_serv* represents how a vulnerability might affect such service in a machine. The details of these two constraints are discussed in

Section 6.1.2. In the system and model design, ILLATION defines the following logic features to compute these two constraints to enhance the existing network agnostic vulnerability prioritization solutions.

Three logic features: ILLATION derives the value of two constraints based on the defined three logic features that represent the overall network reachability, the protocol match, and the vulnerability_service status (details are discussed in Section 6.1.2). The **logic feature of overall network reachability** is to avoid the high complexity of generating all possible attack paths in the existing attack graph-based models. ILLATION utilizes the absorption law of the logic reasoning engine, and computes the overall network reachability instead of path enumeration. This design improvement can support using ILLATION in a large scale network system, where the computation complexity is very high when applying the attack graph-based models. The **logic feature of protocol match** considers the various relationships among protocols, services, and neighbor nodes in a network. A matched protocol pair represents a direct attack path to exploit a vulnerability in the target host (i.e., $Node_{dst}$) by matching the protocol that supports network connection (i.e., $Protocol_r$) and the protocol that supports the vulnerable service program running (i.e., $Protocol_s$). For example, in NP-2, FIN1 is only reachable by MailServer1 ($Protocol_r$=IMAP). The vulnerability CVE-2018-8626 (on FIN1) is associated with Operating Systems Windows10 ($Protocol_s$=no protocol). Although these two protocols do not match, usually this means no direct attack path w.r.t. CVE-2018-8626 on FIN1, ILLATION still considers the potential risk that might come from the available connections between WebServer1 and MailServer1, where the attacker might create an indirect attack path to exploit this vulnerability. Thus, ILLATION reduces this vulnerability risk with a risk factor ($P_m = 0.5$). Cyber admins can adjust this risk factor to other values in implementation. The **logic feature of vulnerability_services status in a machine** shows how a vulnerability might affect a service in a machine. A vulnerability usually is associated with a specific service. The description of a vulnerability illustrates how such service could be harmed by this vulnerability. ILLATION parses the given network profiles to capture a service's status in a machine, and defines whether or not a vulnerability could directly harm such service from the NVD records (The National Institute of Standards and Technology, 2020). ILLATION derives the logic feature of vulnerability_service status for a specific machine in a network by reasoning with this given information.

6.1.2 System and Model Design



Discovered Vuls	NP1 (rank)...	NPk (rank)
CVE-2000-1221	0.9997 (1) ...	0.4998 (4)
CVE-2010-0231	0.7997 (3) ...	0.7997 (2)
....		
CVE-2004-0284	0.1429 (9) ...	0.1429 (8)

Figure 6.3: The Overview of ILLATION System Architecture.

This section presents the overview of ILLATION system architecture with two main functions in Figure 6.3: ① *network profile parsing* embeds the given network profiles as a knowledge base, and ② *prioritization reasoning* function generates the network constraints and then prioritizes vulnerabilities. ILLATION supports a cyber analyst to prioritize the vulnerabilities discovered by vulnerability scanning tools. By using ILLATION, the vulnerability risk is assessed under various attack assumptions w.r.t. network profiles. Finally, ILLATION ranks out the high-risk vulnerabilities to a network profile. As shown in Figure 6.3, ILLATION embeds the given network profiles as a knowledge base through ① *network profile parsing.* In ② *prioritization reasoning*, ILLATION generates the constraint of the network that affects the risk of a vulnerability in a given network, and consolidates these constraints and the initial assessment derived by the initial assessment model.

ILLATION implements logical rules in a logic program ProbLog (De Raedt *et al.*, 2007), which contains a set of ground probabilistic facts F in the form of $p :: f$, and a set of rules R, where p is a probability and f is a ground atom. The rule is an expression in the form $h : -b_1, ..., b_n$, where h is an atom and the b_i are literals. $(: -)$ represents the logical implication $(\leftarrow)$, and the comma $(,)$ represents the logic conjunction $(\wedge)$.

Network Profile Parsing

A network profile can be viewed as an attack scenario of a network, i.e., it describes what defensive system an attacker needs to penetrate for a successful exploitation. The cyber admins can assess vulnerability risk under various attack assumptions by feeding network setups into ILLATION to form the network profiles. All nodes in the network profile are embedded into a knowledge base. Given a network with several services running on machines, ILLATION encodes its network setup as a network profile knowledge base with several logical facts, i.e., *network_reachability*, *network_serv*, and *asset_value*. The network reachability information indicates the service level connections among machines; the network service information indicates the available services on the listed machines.

- Network reachability: denotes the network reachability for all nodes in the network, and is in the form of: $network\_reachability(Node_{src}, Node_{dst}, Protocol_r)$, where $Protocol_r$ supports the network communication between $Node_{src}$ and $Node_{dst}$.

- Network service: denotes a set of services installed on a machine $Node$ in the network as $network\_serv(Node, Service, Protocol_s)$, where $Protocol_s$ supports the operation of this service.

- Asset value: denotes the asset value of a machine in the network as $P_v :: asset\_value(Node)$, where P_v is the machine $Node$'s asset value divided by the upper bound of the asset value range. For example, in a network, asset values range from 1 to 10, the machine $Node$ has a value of 7. Thus, its P_v value is 0.7.

ILLATION encodes and reasons with the given inputs of network setups by following Rule 6.1, where the network topology is identified by the connectivity of nodes (e.g., host, firewall, gateway, etc.) in a network. $node\_connect\_l3(Node_{src}, Node_{dst}, Protocol_r)$ represents connection at the layer-3 (identified by IP address), where the $Protocol_r$ represents the associated protocol to an open port on the $Node_{dst}$. ILLATION also considers the popular protocol-based defense mechanisms, such as the firewall rules and flow rules. For example, $security\_rule(Node_{src}, Node_{dst}, Protocol_r)$ represents firewall rules or flow rules that allow network communications under a specific protocol between two hosts.

$$\begin{aligned} network\_reachability(Node_{src}, Node_{dst}, Protocol_r) :- \\ node\_connect\_l3(Node_{src}, Node_{dst}, Protocol_r), \\ security\_rule(Node_{src}, Node_{dst}, Protocol_r). \end{aligned} \tag{6.1}$$

Prioritization Reasoning

ILLATION's prioritization phase aims at ranking vulnerabilities based on the risk score under different network profiles, i.e., network profiles that are associated with mitigation actions. ILLATION can reason vulnerabilities with the time complexity of $O(n)$, where n is the number of nodes in the network.

The ②*prioritization reasoning* function reasons with the given network profile and the initial assessment. Based on the network profile knowledge base, for each machine in the network, ILLATION can indicate which service exists, how critical the service is, and which vulnerability can be used to attack such service through the *constr_corr_serv* and *constr_critical_serv* constraints. ILLATION evaluates the effects of network profiles on vulnerability risk from the defender's view, where it focuses on analyzing the traffic flows of the destination node ($Node_{dst}$).

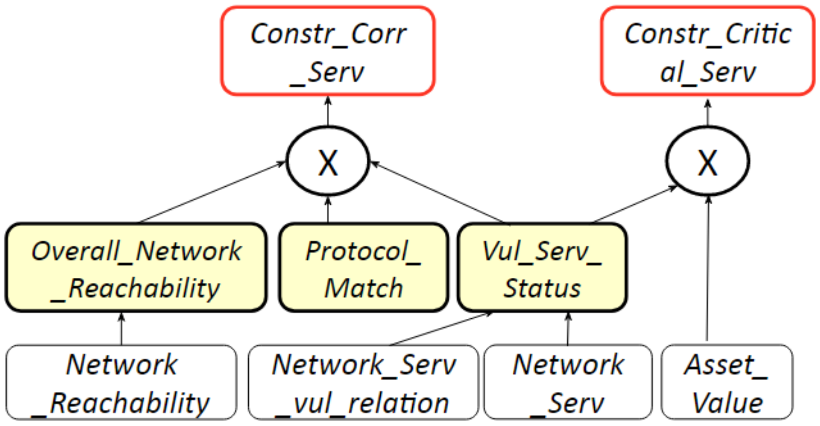


Figure 6.4: The Overview of Network Constraints.

Figure 6.3 presents that ILLATION's prioritization reasoning engine modifies the initial risk assessment with the constraints of *constr_corr_serv* and *constr_critical_serv*. Figure 6.4 shows the overview of these two constraints with three defined logic features. As discussed in Section 6.1.1, ILLATION ranks out high-risk vulnerabilities that could actually affect critical assets. These two constraints range from 0 to 1, which will maintain or reduce the initial assessment of a vulnerability by considering the actual effects of such vulnerability on a machine in the given network profile. Details of these two constraints are discussed in Section 6.1.2. Rule 6.2 presents the prioritization reasoning model as:

$$\begin{aligned} vul\_risk(Cve\_id, Node_{dst}) :- \\ initial\_assess(Cve\_id), \\ constr\_critical\_serv(Cve\_id, Service, Node_{dst}), \\ constr\_corr\_serv(Cve\_id, Service, Node_{dst}). \end{aligned} \tag{6.2}$$

where $initial\_assess(Cve\_id)$ is the output of the initial assessment model. A most-to-least-risky rank-order of vulnerabilities is generated by ② *prioritization reasoning* based on the value of $vul\_risk(Cve\_id, Node_{dst})$. A vulnerability that has the highest risk score will first be recommended for remediation. ILLATION reasons with network profiles for the vulnerability prioritization task without the cyber analyst performing any manual analysis. In the illustrative example, ILLATION outputs the prioritized vulnerabilities listed in Table 6.2 w.r.t the network profiles (NP-1, NP-2) before and after mitigation.

Network Constraints for Prioritization Reasoning

ILLATION ranks out vulnerabilities that both have high risk and could actually affect critical assets as the constraint of $constr\_corr\_serv$ in Rule 6.3 and $constr\_critical\_serv$ in Rule 6.4. In Rule 6.4, the $constr\_critical\_serv$ represents how a vulnerability might affect a service in critical assets. $asset\_value$ represents the critical level of a machine in a network, where a cyber admin must ensure the secure operation of this machine. Usually, a larger value is associated with a higher critical level. To derive these two constraints, ILLATION defines the $overall\_network\_reachability$, $protocol\_match$, and vul_serv_status in logic features 1-3. This section illustrates the details of logic features.

$$\begin{aligned} constr\_corr\_serv(Cve\_id, Service, Node_{dst}) :- \\ overall\_network\_reachability(Node_{dst}, Protocol_r), \\ protocol\_match(Protocol_r, Protocol_s, Node_{dst}), \\ vul\_serv\_status(Cve\_id, Service, Node_{dst}). \end{aligned} \tag{6.3}$$

$$\begin{aligned} constr\_critical\_serv(Cve\_id, Service, Node_{dst}) :- \\ asset\_value(Node_{dst}), \\ vul\_serv\_status(Cve\_id, Service, Node_{dst}). \end{aligned} \tag{6.4}$$

Logic feature 1: overall network reachability. ILLATION generates the overall reachability information of $Node_{dst}$ in the network as $overall\_network\_reachability$ $(Node_{dst}, Protocol_r)$ by reasoning with all existing network reachability in the given network profile for a node $Node_{dst}$ through Rule 6.5. These $network\_reachability$ atoms are viewed as independent events. By following the absorption law, the overall reachability $overall\_network\_reachability$ is computed with the logic disjunction ($\vee$) to all $network\_reachability$ atoms for the node $Node_{dst}$ in Rule 6.5.

$$\begin{aligned} overall\_network\_reachability(Node_{dst}, Protocol_r) :- \\ network\_reachability(Node_{src}, Node_{dst}, Protocol_r). \end{aligned} \tag{6.5}$$

Additionally, to implement Rule 6.1, ILLATION considers the common mitigation actions in a network. In the evaluation, ILLATION uses $node\_connect\_l3(Node_{src},$ $Node_{dst}, Protocol)$ to represent the layer-3 connectivity. By taking layer-3 mitigation actions, such as creating filtering rules, changing IP address, blocking the port, and so on (Chung *et al.*, 2013), the possible attack paths from neighbor hosts to the suspicious host can be restricted. Such mitigation can result in reducing the likelihood of exploiting vulnerabilities associated with layer-3 attacks, e.g., the distributed denial-of-service (DDoS) attack (Zargar *et al.*, 2013).

Logic feature 2: protocol match. ILLATION assesses the correlation between $Protocol_r$ (supporting network communication between hosts) and $Protocol_s$ (supporting the operation of a vulnerable service program on a host) through P_m :: $protocol\_match(Protocol_r, Protocol_s, Node_{dst})$, where P_m is a probabilistic value that can be assigned in implementations. If $Protocol_r$ matches $Protocol_s$, it means the attacker might have a chance to exploit the service's vulnerability through this network connection. A matched protocol pair means a direct attack path exists between two nodes. An unmatched protocol pair only represents that a direct attack path does not exist. But still, the attacker might have a chance to utilize other available connections to create an indirect attack path and then exploit this vulnerability finally. Thus, ILLATION sets the value of P_m as a risk factor to handle this situation. If no matched protocol pair and no direct connection exists, the target node is isolated in this network. In this situation, ILLATION assesses this vulnerability risk as 0 with $P_m = 0$.

To set up the value of P_m, ILLATION considers various relationships among protocols, services, and neighbor nodes in a network, including (1) relationships between protocols on different layers of the network stack, i.e., TCP/IP protocol stack layers. For instance, TCP supports application-layer protocols, such as SMTP, FTP, and so on. ILLATION defines a subset condition (i.e., $TCP \supset \{SMTP, FTP, HTTP, etc.\}$) to represent such layer-based dependency in the reasoning engine; and (2) interactions between neighbor nodes and the target node. ILLATION suggests that if $Protocol_r$ matches $Protocol_s$, P_m is set to 1. Otherwise, if the target node $Node_{dst}$ is still reachable by other neighbor nodes, P_m is set to 0.5; if the $Node_{dst}$ can't be reached by any other nodes in the network, P_m is set to 0. Cyber admins can adjust these values in implementation.

Logic feature 3: the vulnerability_service status in a machine. ILLATION defines *vul_serv_status* to capture how likely a vulnerability might directly harm a service in a machine as Rule 6.6.

$$\begin{aligned} vul\_serv\_status(Cve\_id, Service, Node_{dst}) :- \\ network\_serv(Node_{dst}, Service, Protocol_s), \\ network\_serv\_vul\_relation(Cve\_id, Service). \end{aligned} \tag{6.6}$$

where *network_serv* represents a service's status (running or not exists) in a machine, and *network_serv_vul_relation* represents whether or not a vulnerability can directly harm this service based on records in the national vulnerability dataset (NVD) (The National Institute of Standards and Technology, 2020). For example, 1 :: $network\_serv(webServer, apache, https)$ represents that the Apache service is running in webServer and communicates with other neighbor nodes through the HTTPS protocol. $network\_serv\_vul\_relation(cve-2006-7098, apache)$ indicates that a vulnerability CVE-2006-7098 can directly harm this Apache service based on NVD records. In this way, ILLATION derives the value of $vul\_serv\_status(cve-2006-7098, apache, webServer)$ as 1, which means that the discovered vulnerability CVE-2006-7098 can actually harm the Apache service in webServer. For the network profile, where the Apache service in the webServer is not exists, ILLATION parses this network profile with the value of $network\_serv(webServer, apache, https)$ as 0. Thus, ILLATION derives the value of $vul\_serv\_status(cve-2006-7098, apache, webServer)$ as 0. In this situation, ILLATION reduces the risk assessment for CVE-2006-7098 in webServer to 0.

To assess a vulnerability CVE-2006-7098 in webServer, ILLATION derives the following logic features based on Rule 6.5 and 6.6.

$$\begin{aligned} &overall\_network\_reachability(webServer, https) :- \\ &network\_reachability(vmGroup, webServer, https), \\ &network\_reachability(gatewayServer, webServer, https). \end{aligned}$$

$$\begin{aligned} &vul\_serv\_status(cve-2006-7098, apache, webServer) :- \\ &1 :: network\_serv(webServer, apache, https), \\ &network\_serv\_vul\_relation(cve-2006-7098, apache). \end{aligned}$$

ILLATION also defines the protocol match value $P_m = 1$ as $1 :: protocol\_match$ $(https, https, webServer)$. This is due to the fact that the protocol that supports the network communication for webServer matches the protocol that support Apache service operation. By reasoning with these three logic features, ILLATION then derives the value of $constr\_corr\_serv(cve-2006-7098, apache, webServer)$ as 1 through Rule 6.3, which indicates that the running service Apache in webServer can be reachable by neighbor nodes in the network.

$$\begin{aligned} &constr\_corr\_serv(cve-2006-7098, apache, webServer) :- \\ &1 :: overall\_network\_reachability(webServer, https), \\ &1 :: protocol\_match(https, https, webServer), \\ &1 :: vul\_serv\_status(cve-2006-7098, apache, webServer). \end{aligned}$$

$$constr\_critical\_serv(cve-2006-7098, apache, webServer) :- \\ 1 :: asset\_value(webServer), \\ 1 :: vul\_serv\_status(cve-2006-7098, apache, webServer).$$

Similarly, ILLATION derives the value of $constr\_critical\_serv(cve - 2006 - 7098, apache, webServer)$ as 1 through Rule 6.4, which indicates that the vulnerability CVE-2006-7098 can actually affect the Apache service in webServer.

$1 :: asset\_value(webServer)$ represents that this machine webServer has the highest critical level in this network. When applying mitigation actions in this network by uninstalling the Apache service on webServer, ILLATION derives $0 :: vul\_serv\_status (cve-2006-7098, apache, webServer)$. In this case, $constr\_critical\_serv(cve-2006-7098, apache, webServer)$ is computed with the value of 0, which indicates that the vulnerability CVE-2006-7098 cannot actually affect the Apache service in webServer under this given network profile.

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 in Rule 6.1 and 7.1; 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 specifying a mitigation action that can block assess to such vulnerable services.

Based on Assumption 3, changing one defense rule for mitigating a vulnerability on a node will not impact the *vul_risk* of vulnerabilities on other nodes in the network in Rule 6.2.

6.2 Experimental Evaluation

To evaluate ILLATION, this section studies the performance of the initial assessment model and prioritization reasoning engine, where the initial assessment model decides how accurate the initial assessment of vulnerability risk could be, and the prioritization reasoning engine decides how fast to consolidate the initial risk assessment and the network constraints in various network setups. The performance and scalability of prioritization reasoning engine are evaluated under scalable network setups (50-10k hosts) and various amounts of vulnerabilities (10-10k) in Section 6.2.2. The experimental settings for performance evaluation are illustrated in Section 6.2.1.

6.2.1 Experimental Settings

Vulnerability Mitigation Assumptions

ILLATION takes a new approach by considering the vulnerability mitigation from the defenders' view. It treats a vulnerable end-host (or an application/service) as the attacking target and assumes an attacker may locate inside or outside a network system. In this way, ILLATION can avoid enumerating all possible attack paths, which most of the graph-based vulnerability analysis models require. The cyber analyst performs mitigation actions by using defense mechanisms to effectively reconfigure networks. For example, when using Software Defined Networking (SDN) approaches (Huang *et al.*, 2018), the defense mechanism (e.g., firewall, vulnerability scanning tool, gateway, etc.) can be implemented using an SDN controller to reconfigure network setup and enable/disable basic network flows and access control between hosts and networking devices. ILLATION is capable of using SDN-supported security solutions (e.g., secure service function chaining (Bhamare *et al.*, 2016)) to implement secure networking services (e.g., application firewall, IDS/IPS, DPI, etc.).

Network Setup

A network environment in a private cloud environment is emulated in this evaluation by using Mininet (Mininet, 2021). Mininet is a network simulator that allows users to create a network with virtual hosts, switches (supporting the OpenFlow protocols), controllers, and links. In this experiment evaluation, several scripts are developed to customize network topology and emulate a software-defined network (SDN) (Huang *et al.*, 2018) in our private cloud. Open Network Operating System (ONOS) (Berde *et al.*, 2014) is used as the core SDN controller to support standardized communication protocols. In the real working scenario, the security admin can conduct mitigation actions through a security appliance. All the network profiles that regard layer-2 and layer-3 mitigation actions can be obtained through the SDN controller. Due to the limitation of Mininet and the computer resource on emulating virtual hosts for a network (Di Lena *et al.*, 2021), which is up to 100 hosts in this system. A Python script is developed to simulate the SDN controller data for the large-scale network (i.e., for 500, 1k, 5k, 10k hosts).

To evaluate the performance of the ILLATION reasoning engine on a large-scale network with a large number of vulnerabilities, vulnerabilities deployment is simulated in the network by using a developed Python script. Vulnerabilities are randomly assigned to each host based on operation system compatibility, network service, and opened ports. These vulnerabilities are from 69,730 vulnerabilities associated with the top 50 products in CVEDetails (CVEDetails, 2021). These products cover the common modern operation systems (OS) (i.e., Windows, Linux distributions, and

Mac OS), popular applications, such as web browsers (e.g., Chrome, IE, Edge, Safari, Firefox, etc.), and server software (e.g., databases, web servers, etc.). According to the recent report (EdgeScan, 2020), there are over 75% of applications and over 67% of assets have at least one vulnerability. In this experiment design, each host was assigned 0 to 5 operation system vulnerabilities. There are 0 to 15 applications vulnerabilities associated with various applications discussed above. All the assigned vulnerabilities are recorded as (Cve_id, Node) pairs. ILLATION creates the query as $vul\_risk(Cve\_id, Node)$ based on these pairs.

Model Implementation

ILLATION is implemented on a Linux PC with two Intel(R) Xeon(R) CPU @2.30GHz processors and 26GB RAM. ILLATION is developed in Python 3.7 and the logic programming language ProbLog (De Raedt *et al.*, 2007), with around 2,500 lines of code in total.

In the evaluation, ILLATION adopts LICALITY (Zeng *et al.*, 2021) (discussed in Chapter 5) to develop an neuro-symbolic model in DeepProbLog (Manhaeve *et al.*, 2018) as the initial assessment model. To setup the initial assessment model, ILLATION integrates an NLP tool of latent semantic analysis (Dumais, 2004) to transform vulnerability descriptions to be vector representations. All vulnerability descriptions are embedded as vector representations by using the NLP techniques of Term Frequency-Inverse Document Frequency (TF-IDF) and truncated Singular Value Decomposition (SVD) algorithm. ILLATION captures such latent vulnerability features from its description as LSA feature, and captures the CVSS $\langle AC, C, I, A \rangle$ as CVSS feature by using a developed python script.

ILLATION's assessment model assesses vulnerability risk w.r.t. a network from two measurements: the likelihood of exploitation (w.r.t. the probability of compromise) and the criticality of exploitation (w.r.t. the consequence of compromise) as (Zeng *et al.*, 2021). To learn the features of these two measurements, a neuro-symbolic learning-based assessment model is developed in DeepProbLog (Manhaeve *et al.*, 2018), where the NN side learns the pattern that contributes to assessing the likelihood of exploitation from vulnerability description data in ②, and the symbolic side learns the pattern that contributes to assessing the criticality of exploitation from CVSS metric data. The symbolic side is developed in probabilistic logic programming (PLP) (Manhaeve *et al.*, 2018; De Raedt *et al.*, 2007). ILLATION trains this neuro-symbolic learning-based assessment model by following the parameter learning of DeepProblog (Manhaeve *et al.*, 2018) as 1) the NN model's parameters are updated by back-propagation (Werbos, 1990) with the gradients of the loss w.r.t. the output of the NN side (e.g., $likelihood\_exploit$); and 2) the symbolic side's parameters are updated by gradient semiring (Manhaeve *et al.*, 2018) with the gradients of the loss w.r.t. the learnable probabilities in the symbolic side (e.g., in the $criticality\_exploit$).

To train this initial assessment model, a dataset $\mathcal{D}$ is constructed with the data instance and the $label \in \{0, 1\}$ represents whether a vulnerability is less or more likely to be exploited. The training set, validation set, and testing set from the dataset $\mathcal{D}$ (e.g., at the rate of 80%, 10%, and 10%) are randomly sampled by excluding vulnerabilities for labeling and evaluation that are covered in evaluation cases. ILLATION obtains $initial\_assess(Cve\_id)$ in Rule 6.2 from this initial assessment model and prioritizes vulnerabilities under the given network profiles. In the implementation, a knowledge base is constructed to represent the relationship between vulnerabilities and services as $network\_serv\_vul\_relation(Cve\_id, Service)$. In this evaluation, a developed script randomly deploys vulnerabilities from 69,730 vulnerabilities that are associated with the top 50 products in CVEDetails (CVEDetails, 2021). Thus, the $network\_serv\_vul\_relation$ knowledge base covers these vulnerabilities' relations w.r.t. services in the evaluation.

Training the Risk Assessment Model

ILLATION processes vulnerabilities in the NVD (The National Institute of Standards and Technology, 2020) format. The historical threat contains the exploits in the wild (e.g., ExploitDB (The Offensive Security, 2020)) and historical attack records in the network. We extract vulnerabilities from 1999 June to 2021 November in NVD with 163,404 vulnerabilities in total (The National Institute of Standards and Technology, 2020). All NVD vulnerabilities have the unique Common Vulnerabilities and Exposures identification number (CVE-ID). Some CVEs are rejected due to a duplicated record or are reserved for reports in the future. By excluding invalid CVEs marked as “disputed” or “unsupported when assigned” in descriptions, we removed 971 CVE_ID in the NVD and refined it to 162,433 vulnerabilities in this study. We develop a parser in Python to extract vulnerabilities’ CVE_IDs, descriptions, and CVSS vector from JSON files (The National Institute of Standards and Technology, 2020). In the evaluation, we first clean vulnerability descriptions data by transforming 162,433 vulnerabilities’ descriptions into a text corpus through the most popular data clean processes, such as lower-casing all text data, removing stop-words, stemming, and normalizing words. Then, the vulnerability description is embedded as a vector representation (called LSA feature) by using the Sklearn (Scikit-learn, 2021) toolkits. In this study, we explore the different combinations of dimensions for the LSA feature. In the evaluation, we select the dimension of the LSA feature as 150, which has the best explained variance ratio as 0.87.

Due to the lack of attack datasets that have the associated network system environmental data, the historical attack records is simulated based on the real APT attack reported by the cybersecurity company FireEye in 2017 (The FireEye, 2017).

The APT attack is a stealthy threat that uses continuous and sophisticated hacking techniques to access a system over months or years and is very challenging to be detected (Alshamrani *et al.*, 2019). Table 6.3 presents the details of the threat.

Table 6.3: The Overview of Threat Features in the Evaluation

Case	The Software Set (S_{sw})	The CVSS Set[*] (S_{CVSS})
The FireEye (2017)	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.

The neural network side of the neuro-symbolic learning-based assessment model is a four-layer neural network model with 150 nodes in the input layer, 100 nodes in the first hidden layer, 50 nodes in the second hidden layer. A dropout layer is applied with a 0.25 dropout rate to prevent overfitting. To train the assessment model, the training set, validation set, and testing set are randomly sampled from the dataset at the rate of 80%, 10%, and 10% by excluding vulnerabilities for labeling and evaluation.

6.2.2 Results

ILLATION consolidates the initial risk assessment derived by the initial assessment model and the network constraints. These constraints are derived from network setups by the prioritization reasoning engine. The network setups are 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. A logical fact in the network profile knowledge base represents the network traffic reachability between hosts. For example, the logic fact of $network\_reachability(host1, host2, http)$ indicates that host1 can send packets to host2 through HTTP. ILLATION can identify network reachability representations at both layer-2 and layer-3 by associating a host to its MAC address and IP address, respectively. ILLATION can execute on multiple network profile knowledge bases and can output the vulnerability prioritization result to a specific network profile. By reasoning with the network profile knowledge base and the initial assessment model's output, the prioritization reasoning engine assesses vulnerability risk as Rule 6.2.

Table 6.4: The Running Time (in Second) of the Network Profiling Parser in a Scalable Computer Networking 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

The running time of the network profiling parser and the prioritization reasoning engine are measured to show ILLATION's performance and scalability in a scalable network environment. This scalable network environment has vulnerable hosts ranging from 10 to 10k and have up to 10k vulnerabilities. The average node connectivity w.r.t. a host is around 31 in the evaluation. Table 6.4 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 6.4 within 5 seconds. The running time of network profiling parser for a large size network setup data (12.65 GB) is measured as well, and this running time is around 622 seconds.

Table 6.5: The Running Time (in Second) of the Prioritization Reasoning Engine for Assessing Vulnerabilities in a Scalable Computer Networking System

Vulnerable Hosts	10 Vulnerabilities	50 Vulnerabilities	100 Vulnerabilities	500 Vulnerabilities	1K Vulnerabilities	5K Vulnerabilities	10K Vulnerabilities
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 6.5, the reasoning engine's running time is affected by the number of vulnerable hosts and the number of vulnerabilities. The running time of the prioritization reasoning engine increases as 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 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, Figure 6.5 presents the average running time for assessing a vulnerability. Due to space limits, this chart only labels the average running time of assessing 10, 50, 100, and 10000 vulnerabilities. Figure 6.5 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 vul-

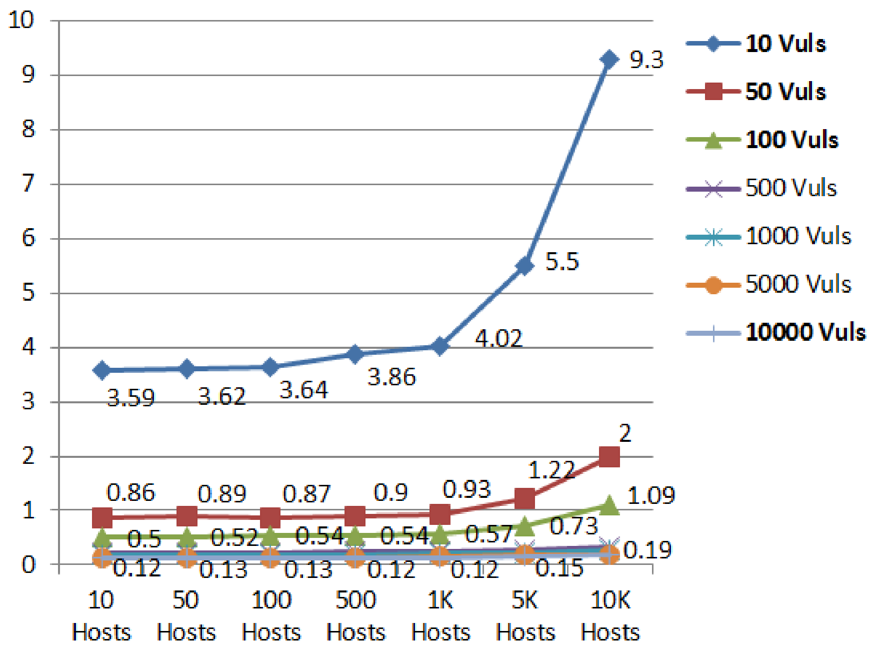


Figure 6.5: The Average Running Time (in Second Per Vulnerability) for Assessing a Vulnerability in the Evaluation

nerabilities, the average running time per vulnerability changes from 0.12 seconds (in 10 hosts) to 0.19 seconds (in 10k hosts). While, for 10 vulnerabilities, it changes from 3.59 seconds (in 10 hosts) to 9.3 seconds (in 10k hosts). ILLATION implements Rule 6.3, 6.4, and 6.5, where more vulnerable hosts usually associate with more network profile data (i.e., network reachability, network service, network asset). 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 6.4. 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) (Darwiche, 2011), which is a tree structure that defines the logical and probabilistic relationships among nodes for reasoning. So, 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 6.5 also shows 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. This finding supports that ILLATION has the potential to handle the vulnerability prioritization tasks in a large-scale network.

6.3 Summary

In this chapter, ILLATION is proposed, which is a comprehensive framework for vulnerability prioritization in the scalable working scenarios. ILLATION enhances the existing network agnostic vulnerability prioritization approaches by reasoning with real network system setups (network profiles). Two network constraints are derived based on the given network profiles. ILLATION consolidates the initial risk assessment and network constraints as an overall risk assessment by a prioritization reasoning engine. ILLATION models the overall risk of a vulnerability w.r.t. a network and security setups, and establishes an efficient and scalable solution to assess a large number of vulnerabilities in the scalable networking system.

Chapter 7

CONCLUSION AND FUTURE WORK

This chapter summarizes the key contributions of this dissertation and present a brief discussion on promising future research directions.

7.1 Conclusion

In this dissertation, a risk-based vulnerability prioritization framework is proposed. In this framework, a vulnerability risk is assessed based on the characterized vulnerability risk features (discussed in Chapter 4). This framework contains two stages by successfully incorporating multiple vulnerability attributes and the network and security setups into vulnerability prioritization. These two stages include the vulnerability risk assessment (discussed in Chapter 5) and the vulnerability prioritization (discussed in Chapter 6). This framework employs novel combination of neuro-symbolic learning, natural language processing and logical reasoning techniques that model the overall risk of a vulnerability w.r.t. a network profile. The main contributions are as follows.

Characterizing vulnerability risk: This proposed framework characterizes vulnerability risk from the likelihood of exploitation and the criticality of exploitation. The initial risk assessment is refined by two network constraints, which are derived from the given network profile based on three defined logic features as overall network reachability, protocol match and vulnerability_services status in a machine. By reasoning with both the node connectivity and the defense rules from a network profile, this framework considers which service exists, how critical the service is, and which vulnerability can be used to attack such service.

Introducing the neuro-symbolic computing to vulnerability prioritization: Based on our best knowledge, the proposed vulnerability risk assessment model is the first study to assess vulnerability risk by utilizing the neuro-symbolic computing. This proposed vulnerability risk assessment model learns threat attributes, and provides a realistic assessment solution based on past risk/attack history for a computer networking system. The output of the symbolic side (based on logic reasoning) refines the output of the neural network model (based on machine learning/deep learning). By reasoning with the likelihood of exploitation from LSA features and the criticality of exploitation from the CVSS features, this neuro-symbolic computing-based risk assessment model is more data-efficient than the neural network-based model.

Establishing an efficient and scalable solution to assess a large number of vulnerabilities: This framework embeds the network setup data as knowledge bases. To avoid the high computation complexity in the existing network specific approach, the absorption law of logic reasoning engine is applied by deriving the overall network reachability for the host instead of path enumeration. In the evaluation, the proposed vulnerability prioritization reasoning engine can assess 1k vulnerabilities in the network (up to 10k vulnerable hosts), in about 4.5 minutes in total, which could support the network security operation in real-time. To assess vulnerabilities under the same network setups, the average running time of assessing 10k vulnerabilities is always at the lowest level with up to 0.19 seconds per vulnerability. This makes the proposed vulnerability prioritization reasoning engine suitable for implementation in a large-scale network system, which usually has a large number of vulnerabilities.

7.2 Future Work

In the evaluation, due to the lack of attack data and network setups data from a real system, the proposed vulnerability risk assessment model is evaluated with historical threats (identified by Microsoft vulnerabilities in MVs_2015 and APT attack vulnerabilities in APTVs_2017). 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. Furthermore, the proposed prioritization reasoning engine is evaluated in an emulated computer networking system with randomly deployed vulnerabilities simulated by a script. The overall result shows the potential to handle vulnerability prioritization tasks in large-scale networks with a large number of vulnerabilities. To implement this prioritization reasoning engine in a real working scenario, users might need to customize the basic logic engines w.r.t. their own network setup features. Additionally, only the basic defense rules are applied when implementing the prioritization reasoning engine in this dissertation. As discussed in Chapter 6, this prioritization reasoning engine can be extended to analyze more complex scenarios with more input features. Thus, the future work can investigate the performance of the proposed framework by deploying it in a real large-scale network system to test its performance on vulnerability management.

By considering the access control policy in the reasoning engine, this prioritization reasoning engine can be implemented as Rule 7.1, where $acl(Node_{src}, Node_{dst}, Priv)$ represents the access control policy. According to the existing study (Ou *et al.*, 2005), a successful vulnerability exploit can be indicated by the escalation of privilege. By defining the access control policy with accounts (e.g., with the client, user, root privilege, etc.), this reasoning engine can extend Rule 6.1 to Rule 7.1 to capture the change of privilege on machines. Additionally, this reasoning engine can extend to use $node\_connect\_l2(Node_{src}, Node_{dst})$ to represent the layer-2 connectivity to assess vulnerabilities for layer-2 attacks (e.g., the ARP Spoofing attack). In this way, the proposed framework can be used to analyze layer-2 mitigation actions, which adjust network configurations at the data link layer (e.g., isolating a suspicious host by changing its MAC address (Chung *et al.*, 2013)). The good extension capability of the logic model has been proved in other logic model-based studies (Bacic *et al.*, 2006). By building upon this basic model, this framework has the capacity to analyze more complex scenarios with more input features in future studies.

$$\begin{aligned} network\_reachability(Node_{src}, Node_{dst}, Protocol, Priv) :- \\ node\_connect\_l2(Node_{src}, Node_{dst}), \\ node\_connect\_l3(Node_{src}, Node_{dst}, Protocol), \\ acl(Node_{src}, Node_{dst}, Priv), \\ security\_rule(Node_{src}, Node_{dst}, Protocol). \end{aligned} \tag{7.1}$$

In addition, although this framework proposes the neuro-symbolic computing-based assessment model, it can be easily extended to other assessment models (e.g., machine learning-based models). This is because the prioritization reasoning engine processes the initial assessment value from the assessment model as a given input, and outputs the consolidated assessment for a vulnerability w.r.t. the given network profile. The future work can extend this framework to various vulnerability assessment models. Furthermore, the future study can involve the attack detection/prevention tools (e.g., Intrusion Detection/Prevention System, Deep Package Inspection, etc.), address how to detect and handle the attack traffic, and investigate how to generate an optimal mitigation action plan as well.

REFERENCES

Aboud, J., "Prioritizing vulnerabilities: Why it's essential to an effective vulnerability management program", URL `https://www.ivanti.com/blog/prioritizing-vulnerabilities`, last accessed 21 April 2021 (2019).

Allodi, L. and S. Etalle, "Towards realistic threat modeling: attack commodification, irrelevant vulnerabilities, and unrealistic assumptions", in "Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense", pp. 23–26 (2017).

Allodi, L. and F. Massacci, "Comparing vulnerability severity and exploits using case-control studies", ACM Transactions on Information and System Security (TISSEC) **17**, 1, 1–20 (2014).

Allodi, L. and F. Massacci, "Attack potential in impact and complexity", in "Proceedings of the 12th International Conference on Availability, Reliability and Security", pp. 1–6 (2017).

Allodi, L., F. Massacci and J. Williams, "The Work-Averse Cyberattacker Model: Theory and Evidence from Two Million Attack Signatures", Risk Analysis (2021).

Allodi, L., F. Massacci and J. M. Williams, "The work-averse cyber attacker model: Theory and evidence from two million attack signatures", Available at SSRN 2862299 (2017).

Alperin, K., A. Wollaber, D. Ross, P. Trepagnier and L. Leonard, "Risk prioritization by leveraging latent vulnerability features in a contested environment", in "Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security", pp. 49–57 (2019).

Alshamrani, A., S. Myneni, A. Chowdhary and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities", IEEE Communications Surveys & Tutorials **21**, 2, 1851–1877 (2019).

Bacic, E., M. Froh and G. Henderson, "Mulval extensions for dynamic asset protection", Tech. rep., CINNABAR NETWORKS INC OTTAWA (ONTARIO) (2006).

Bao, T., R. Wang, Y. Shoshitaishvili and D. Brumley, "Your exploit is mine: Automatic shellcode transplant for remote exploits", in "2017 IEEE Symposium on Security and Privacy (SP)", pp. 824–839 (IEEE, 2017).

Berde, P., M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os", in "Proceedings of the third workshop on Hot topics in software defined networking", pp. 1–6 (2014).

Bhamare, D., R. Jain, M. Samaka and A. Erbad, "A survey on service function chaining", Journal of Network and Computer Applications **75**, 138–155 (2016).

Bozorgi, M., L. K. Saul, S. Savage and G. 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”, pp. 105–114 (2010).

Bromander, S., A. Jøsang and M. Eian, “Semantic cyberthreat modelling.”, in “STIDS”, pp. 74–78 (2016).

Bullough, B. L., A. K. Yanchenko, C. L. Smith and J. 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”, pp. 45–53 (2017).

Chang, K. W., S. W. Yih and C. Meek, “Multi-Relational Latent Semantic Analysis”, in “Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)”, (Association for Computational Linguistics, 2013).

Chen, H., R. Liu, N. Park and V. Subrahmanian, “Using twitter to predict when vulnerabilities will be exploited”, in “Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining”, pp. 3143–3152 (2019).

Chung, C.-J., P. Khatkar, T. Xing, J. Lee and D. Huang, “Nice: Network intrusion detection and countermeasure selection in virtual network systems”, IEEE transactions on dependable and secure computing **10**, 4, 198–211 (2013).

CVEDetails, “Top 50 products by total number of ”distinct” vulnerabilities”, URL `https://www.cvedetails.com/top-50-products.php`, last accessed 15 Oct 2021 (2021).

Darwiche, A., “Sdd: A new canonical representation of propositional knowledge bases”, in “Twenty-Second International Joint Conference on Artificial Intelligence”, (2011).

De Raedt, L., A. Kimmig and H. Toivonen, “Problog: A probabilistic prolog and its application in link discovery.”, in “IJCAI”, vol. 7, pp. 2462–2467 (Hyderabad, 2007).

De Raedt, L., R. Manhaeve, S. Dumancic, T. Demeester and A. Kimmig, “Neurosymbolic= neural+ logical+ probabilistic”, in “NeSy’19@ IJCAI, the 14th International Workshop on Neural-Symbolic Learning and Reasoning”, (2019).

Dempsey, K., E. Takamura, P. Eavy and G. Moore, “Automation support for security control assessments: Software vulnerability management”, Tech. rep., National Institute of Standards and Technology (2020).

Di Lena, G., A. Tomassilli, D. Saucez, F. Giroire, T. Turletti and C. Lac, “Distrinet: a mininet implementation for the cloud”, ACM SIGCOMM Computer Communication Review **51**, 1, 2–9 (2021).

Duan, C., Z. Wang, H. Ding, M. Jiang, Y. Ren and T. Wu, "A vulnerability assessment method for network system based on cooperative game theory", in "International Conference on Algorithms and Architectures for Parallel Processing", pp. 391–398 (Springer, 2019).

Dumais, S. T., "Latent semantic analysis", Annual review of information science and technology **38**, 1, 188–230 (2004).

EdgeScan, "2020 Vulnerability Statistics Report", URL `https://cdn2.hubspot.net/hubfs/4118561/BCC030%20Vulnerability%20Stats%20Report%20(2020)_WEB.pdf`, last accessed 25 Nov 2021 (2020).

Edkrantz, M., S. Truvé and A. Said, "Predicting vulnerability exploits in the wild", in "2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing", pp. 513–514 (IEEE, 2015).

Farris, K. A., A. Shah, G. Cybenko, R. Ganesan and S. Jajodia, "Vulcon: A system for vulnerability prioritization, mitigation, and management", ACM Transactions on Privacy and Security (TOPS) **21**, 4, 1–28 (2018).

Feutrill, A., D. Ranathunga, Y. Yarom and M. Roughan, "The effect of common vulnerability scoring system metrics on vulnerability exploit delay", in "2018 Sixth International Symposium on Computing and Networking (CANDAR)", pp. 1–10 (IEEE, 2018).

Figueroa-Lorenzo, S., J. Añorga and S. Arrizabalaga, "A survey of iiot protocols: A measure of vulnerability risk analysis based on cvss", ACM Computing Surveys (CSUR) **53**, 2, 1–53 (2020).

Garcez, A. d., 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", arXiv preprint arXiv:1905.06088 (2019).

Huang, D., A. Chowdhary and S. Pisharody, *Software-Defined networking and security: from theory to practice* (CRC Press, 2018).

Jack Wallen, "CVSS Struggles to Remain Viable in the Era of Cloud Native Computing", URL `https://thenewstack.io/cvss-struggles-to-remain-viable-in-the-era-of-cloud-native-computing/`, last accessed 21 May 2021 (2020).

Jacobs, J., S. Romanosky, I. Adjerid and W. Baker, "Improving vulnerability remediation through better exploit prediction", Journal of Cybersecurity **6**, 1 (2020).

Jajodia, S., S. Noel and B. O'berry, "Topological analysis of network attack vulnerability", in "Managing cyber threats", pp. 247–266 (Springer, 2005).

Johnson, P., "What is cvss v3.1? understanding the new cvss", URL `https://resources.whitesourcesoftware.com/blog-whitesource/understanding-cvss-v3-1n`, last accessed 1 February 2021 (2019).

Kaplan, S. and B. J. Garrick, “On the quantitative definition of risk”, Risk analysis **1**, 1, 11–27 (1981).

Kingma, D. P. and J. Ba, “Adam: A method for stochastic optimization”, arXiv preprint arXiv:1412.6980 (2014).

Lai, Y.-P. and P.-L. Hsia, “Using the vulnerability information of computer systems to improve the network security”, Computer Communications **30**, 9, 2032–2047 (2007).

Lyon, G. F., *Nmap network scanning: The official Nmap project guide to network discovery and security scanning* (Insecure. Com LLC (US), 2008).

Manhaeve, R., S. Dumancic, A. Kimmig, T. Demeester and L. De Raedt, “Deepproblog: Neural probabilistic logic programming”, in “Advances in Neural Information Processing Systems”, pp. 3749–3759 (2018).

Mateski, M., C. M. Trevino, C. K. Veitch, J. Michalski, J. M. Harris, S. Maruoka and J. Frye, “Cyber threat metrics”, Sandia National Laboratories p. 30 (2012).

McQueen, M. A., W. F. Boyer, M. A. Flynn and G. A. Beitel, “Time-to-compromise model for cyber risk reduction estimation”, in “Quality of protection”, pp. 49–64 (Springer, 2006).

Mell, P., K. Scarfone and S. Romanosky, “Common vulnerability scoring system”, IEEE Security & Privacy **4**, 6, 85–89 (2006).

Mell, P., K. Scarfone and S. 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”, vol. 1, p. 23 (2007).

Microsoft, “Microsoft vulnerabilities report 2020”, URL `https://www.beyondtrust.com/resources/whitepapers/microsoft-vulnerability-report`, last accessed 8 March 2021 (2020).

Mininet, “Mininet:an instant virtual network on your laptop (or other pc)”, URL `http://mininet.org/`, last accessed 15 Oct 2021 (2021).

Miura-Ko, R. A. and N. Bambos, “Securerank: A risk-based vulnerability management scheme for computing infrastructures”, in “2007 IEEE International Conference on Communications”, pp. 1455–1460 (IEEE, 2007).

Nikolai Mansourov, “Ranking weakness findings”, URL `https://www.oasis-open.org/committees/download.php/62624/Ranking_weakness_findings.pdf`, last accessed 21 May 2021 (2018).

Oltsik, J., “Vulnerability Management Woes Continue but There Is Hope”, URL `https://www.esg-global.com/blog/vulnerability-management-woes-continue-but-there-is-hope`, last accessed 5 April 2021 (2019).

Ou, X., S. Govindavajhala, A. W. Appel *et al.*, "Mulval: A logic-based network security analyzer.", in "USENIX security symposium", vol. 8, pp. 113–128 (Baltimore, MD, 2005).

Pytorch, "Early Stopping", URL `https://pytorch-lightning.readthedocs.io/en/latest/common/early_stopping.html`, last accessed July 1 2021 (2021).

Rapid7 Corp., "Vulnerability Remediation vs. Mitigation: What's the Difference?", URL `https://www.rapid7.com/blog/post/2020/09/14/vulnerability-remediation-vs-mitigation-whats-the-difference/`, last accessed 15 Oct 2021 (2020).

Ross, D. M., A. B. Wollaber and P. C. Trepagnier, "Latent feature vulnerability ranking of cvss vectors", in "Proceedings of the Summer Simulation Multi-Conference", pp. 1–12 (2017).

Sabottke, C., O. Suciu and T. Dumitraș, "Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits", in "24th {USENIX} Security Symposium ({USENIX} Security 15)", pp. 1041–1056 (2015).

Scikit-learn, "sklearn api", URL `https://scikit-learn.org/stable/`, last accessed 15 Oct 2021 (2021).

Shah, A., K. A. Farris, R. Ganesan and S. Jajodia, "Vulnerability selection for remediation: An empirical analysis", The Journal of Defense Modeling and Simulation p. 1548512919874129 (2019).

Sheyner, O., J. Haines, S. Jha, R. Lippmann and J. M. Wing, "Automated generation and analysis of attack graphs", in "Proceedings 2002 IEEE Symposium on Security and Privacy", pp. 273–284 (IEEE, 2002).

Spring, J., A. Hatleback, A. Manion and D. Shic, "Towards improving cvss", SEI, CMU, Tech. Rep (2018).

Sun, N., J. Zhang, P. Rimba, S. Gao, L. Y. Zhang and Y. Xiang, "Data-driven cybersecurity incident prediction: A survey", IEEE Communications Surveys & Tutorials **21**, 2, 1744–1772 (2018).

The Cybersecurity & Infrastructure Security Agency (CISA), "Apt actors chaining vulnerabilities against sltt, critical infrastructure, and elections organizations", URL `https://us-cert.cisa.gov/ncas/alerts/aa20-283a`, last accessed 1 July 2021 (2020).

The Cybersecurity and Infrastructure Security Agency, "Top 30 targeted high risk vulnerabilities", URL `https://us-cert.cisa.gov/ncas/alerts/`, last accessed 7 March 2021 (2015).

The Cybersecurity and Infrastructure Security Agency, "Dams sector cybersecurity capability maturity model (c2m2)", URL `https://www.cisa.gov/sites/default/files/publications/dams-c2m2-508.pdf`, last accessed 9 December 2020 (2016).

The Cybersecurity and Infrastructure Security Agency, “Top 10 most exploited vulnerabilities 2016–2019”, URL https://us-cert.cisa.gov/ncas/alerts/, last accessed 3 August 2020 (2020).

The FireEye, “APT28: At the Center Of the Storm”, URL https://www2.fireeye.com/rs/848-DID-242/images/APT28-Center-of-Storm-2017.pdf, last accessed July 1 2021 (2017).

The Forum of Incident Response and Security Teams, “Common vulnerability scoring system v3.0 calculator”, URL https://www.first.org/cvss/calculator/3.0, last accessed 9 December 2020 (2015).

The Forum of Incident Response and Security Teams, “Common Vulnerability Scoring System SIG”, URL https://www.first.org/cvss/, last accessed 1 July 2021 (2021).

The National Institute of Standards and Technology, “NVD: National Vulnerability Database”, URL https://nvd.nist.gov/general, last accessed 25 August 2020 (2020).

The Offensive Security, “Exploitdb”, URL https://www.exploit-db.com/, last accessed 25 August 2020 (2020).

The Open Web Application Security Project, “Threat modeling”, URL https://owasp.org/www-community/Threat_Modeling, last accessed 3 March 2021 (2021).

Tupper, M. and A. N. Zincir-Heywood, “Vea-bility security metric: A network security analysis tool”, in “2008 Third International Conference on Availability, Reliability and Security”, pp. 950–957 (IEEE, 2008).

Verizon Corp., “Data breach investigation report (2021)”, URL https://www.verizon.com/business/resources/reports/dbir/2021/results-and-analysis/, last accessed 15 Oct 2021 (2021).

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

Wiki, “Stochastic gradient descent”, URL https://en.wikipedia.org/wiki/Stochastic_gradient_descent, last accessed 4 April 2021 (2020).

Wiki, “Nessus”, URL https://en.wikipedia.org/wiki/Nessus_(software), last accessed 8 June 2021 (2021).

Woollacott, E., “Measuring risk: Organizations urged to choose defense-in-depth over CVE whack-a-mole”, URL https://www.ivanti.com/blog/prioritizing-vulnerabilities, last accessed 30 May 2021 (2021).

Yadav, G. and K. Paul, “Patchrank: Ordering updates for scada systems”, in “2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)”, pp. 110–117 (IEEE, 2019).

Zargar, S. T., J. Joshi and D. Tipper, “A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks”, IEEE communications surveys & tutorials **15**, 4, 2046–2069 (2013).

Zeng, Z., Z. Yang, D. Huang and C.-J. Chung, “LICALITY–Likelihood and Criticality: Vulnerability Risk Prioritization Through Logical Reasoning and Deep Learning”, IEEE Transactions on Network and Service Management, early access (2021).

Zimmermann, T., N. Nagappan and L. Williams, “Searching for a needle in a haystack: Predicting security vulnerabilities for windows vista”, in “2010 Third International Conference on Software Testing, Verification and Validation”, pp. 421–428 (IEEE, 2010).