2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-1658-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/TRUSTCOM53373.2021.00137

Multi-label Positive and Unlabeled Learning and its Application to Common Vulnerabilities and Exposure Categorization

Masaki Aota [1] †, Tao Ban*, Takeshi Takahashi*, Noboru Murata†,
*National Institute of Information and Communications Technology, Tokyo, Japan
†Waseda University, Tokyo, Japan
masaki.aota@ruri.waseda.jp, bantao@nict.go.jp, takeshi_takahashi@ieee.org, noboru.murata@eb.waseda.ac.jp

***Abstract*—The widely adopted Common Weakness Enumeration (CWE), which stores and manages software and hardware vulnerability reports known as Common Vulnerabilities and Exposures (CVE) in a hierarchical structure, provides common baseline standard for weakness identification, mitigation, and prevention efforts. In this paper, we propose a machine-learning based method to assign pertinent CWE identifiers to new CVE entries. The proposed method formulates the task as a multi-label classification problem and exploits positive and unlabeled learning to address the lack of multi-labelled samples in learning. In evaluations, the proposed method demonstrated preferable performance compared to traditional multi-label classifiers. In particular, case studies demonstrated that multiple CWE identifiers assigned to CVE entries carry essential information that can benefit security practices.**

I. Introduction

Due to their potential to cause serious incidents and heavy loss, the management of software and hardware vulnerabilities is essential in modern society. Properly classifying known vulnerabilities helps us identify the past trends and prevalence of vulnerabilities, which can facilitate quick reactions when similar vulnerabilities occur.

The National Vulnerability Database (NVD) [1] is the most widely adopted database for managing software and hardware vulnerabilities. In the NVD, vulnerabilities are organized as Common Vulnerabilities and Exposures (CVE) entries, and each CVE entry is assigned a Common Weakness Enumeration (CWE) [2] identifier as an indicator of its category. CWE is currently used as a common language to discuss security weaknesses in software and hardware development, and it is used a common baseline standard for various security products. Searching the NVD database with a CWE identifier (CWE-ID) will direct you to a webpage that provides detailed information about the vulnerability, including potential damage and guidelines for identification, mitigation, and prevention.

CWE is often viewed in a hierarchical structure to illustrate the tree-like relationships among the weaknesses that exist at different levels of abstraction. Due to this hierarchical structure, the CWE-IDs assigned to CVE entries inherently contain associative information at multiple levels.

[1] This work is based on research conducted by Masaki Aota during his time as a research assistant at National Institute of Information and Communications Technology, Japan.

For example, a CVE assigned with CWE-521 (*weak password requirements*) is also associated with CWE-200 (*exposure of sensitive information to an unauthorized actor*) because CWE-521 is a sub-node of CWE-200 in the hierarchical structure. To prevent confusion that may rise when assigning a CVE entry with many CWE-IDs in the same lineage, analysts tend to associate it with only a single pertinent CWE-ID even if it fits with multiple CWE-IDs. However, due to the limitation of using a hierarchical structure to model the complicated relationships among weaknesses, we have found many cases where CVE entries assigned with a single CWE-ID match CWE-IDs in different lineages. Take CVE-2008-3550 as an example. In the NVD, CVE-2008-3550 is assigned to CWE-200 (*exposure of sensitive information to an unauthorized actor*). However, CWE-79 (*improper neutralization of input during web page generation ("cross-site scripting")*) is also appropriate because "possibly related to a cross-site scripting (XSS) vulnerability" appears in the CVE description. In the CWE hierarchy, CWE-200 and CWE-79 belong to different sub-branches with no common progenitor.

CVE-2008-3550

The CQWeb login page in IBM Rational ClearQuest 7.0.1 allows remote attackers to **obtain potentially sensitive information** (page source code) via a combination of ?script? and ?/script? sequences in the id field, **possibly related to a cross-site scripting (XSS) vulnerability.** [3]

To further improve the reliability of CWE-IDs assigned to CVEs in the NVD and provide accurate and relevant information to security professionals, we propose a machine learning-based method to assign relevant CWE-IDs to CVE entries. In the proposed method, we adapt Positive and Unlabeled Learning (PU learning) [4] to explore the relationships among CVE entries with all possible CWE-IDs such that CVE entries are assigned multiple CWE-IDs that match their descriptions. The proposed method demonstrates preferable performance compared to traditional multi-label learning on a CVE dataset that was labelled manually with multiple CWE-IDs by security experts.

The main contributions of this paper are summarized as

2324-9013/21/$31.00 ©2021 IEEE
DOI 10.1109/TrustCom53373.2021.00137

follows.

- We propose a machine learning-based method to automate the process of assigning associative CWE-IDs to CVE entries.
- By formulating the task as a multi-label classification problem, the proposed method can assign multiple relevant CWE-IDs to CVE entries, which is a significant complement to the NVD database.
- By adopting PU learning, we address the lack of multi-labelled samples in the learning, which significantly improves prediction accuracy.
- We propose an evaluation criterion to measure the performance of the proposed multi-label classification method.
- We validate the proposed method based on solid numerical results.
- We present a case study of CVE entries assigned with multiple CWE-IDs to validate the inference results in a qualitative manner.

The remainder of this paper is organized as follow. Section II reviews related work. Section III introduces the proposed method, and Section IV describes the experimental settings and proposed evaluation criterion. Section V reports the experimental results, and Section VI presents the case study to investigate CVE entries assigned with multiple CWE-IDs. Finally, conclusions are drawn in Section VII.

II. Related work

Many studies have attempted to automatically assign label information to a vulnerability description using machine learning techniques [5]–[12]. In particular, Na et al., Aota et al., and Aghaei et al. attempted to assign CWE-IDs. Na et al. adapted machine learning for 10 frequently occurring CWE-IDs, and they demonstrated the usefulness of using machine learning [10]. Aota et al. further improved classification performance by expanding to 19 additional useful CWE-IDs [11]. These two studies were based on a system that outputs a single CWE-ID for one CVE entry. Aghaei et al. focused on the hierarchical relationship of CWE-IDs to output a specific CWE-ID and an abstract CWE-ID that encompasses the specific CWE-ID [12] Their system can output multiple CWE-IDs in a conceptual comprehension relationship; however, it cannot handle cases where multiple CWE-IDs of different concepts are mentioned.

Previous studies on multi-label discrimination considering missing labels include Bucak et al. [13] and Kanehira et al. [14]. However, both of these studies designed the risk function based on the rank loss, which may not work effectively for our data because most samples in our data are only single-label samples. In addition, the rank loss incurs $O(C^2)$ computational complexity for the number of classes C; thus, the computation speed is reduced significantly as the number of target classes increases.

The idea of the proposed method, which uses PU learning with the one-vs-rest strategy, is the same as that presented by Kong et al. [15]. However, the method proposed by Kong et al. limits the model to logistic regression. In addition, the method used by Kong et al. to estimate π_t is old (proposed by Elkan et al. [16]) and is known to yield low accuracy. In this study, we updated the method to directly minimize the Pearson divergence, and we introduced a framework to minimize the PU risk, which allows us to use any available model.

III. Proposed Method

A. Concept and Notations

We have devised a method to reinterpret the teacher signal during training. The interpretation of the teacher signal with missing labels is shown in Figure 1. In traditional machine learning, the absence of a label of class t is interpreted as "not belonging to class t." However, in our data, the absence of a label for class t means that "the belongingness to class t is unknown." To solve this gap, we adapt PU learning [4] for each class.

In the following, we define the notations used in this paper. The i-th sample in the dataset is denoted as $\mathbf{x}_i$, ($\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^m$). Here, $\mathcal{X}$ is the defining region of $\mathbf{x}$. The true teacher signal corresponding to the i-th sample is $\mathbf{y}_i^* \in \{-1, +1\}^C$, and the observed teacher signal is $\tilde{\mathbf{y}}_i \in \{0, +1\}^C$, where C is the number of classes in the dataset. $[\mathbf{y}_i^*]_t = -1$ means that the i-th sample does not belong to class t, and $[\mathbf{y}_i^*]_t = +1$ means that the i-th sample belongs to class t. $[\tilde{\mathbf{y}}_i]_t = 0$ represents cases where it is unknown if the i-th sample belongs to class t. In other words, if the t-th element of the teacher signal is $-1/0/+1$, then the sample is taken as negative/unlabeled/positive with respect to class t.

A binary classifier is denoted as $g(\mathbf{x}; \boldsymbol{\theta}) : \mathbb{R}^m \to \mathbb{R}$. Here, if $g(\mathbf{x}; \boldsymbol{\theta}) > 0$, $\mathbf{x}$ is predicted as positive; otherwise, it is predicted as negative. Learning is a procedure to search for a $\boldsymbol{\theta}$ that yields as few misclassifications as possible based on the given data. Take a linear discriminator as an example of a binary classifier: $g(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$.

We define the loss function as $\ell(y, g) : \mathbb{R}^2 \to \mathbb{R}$, and this function gives a large loss if teacher y and prediction g differ significantly. For example, when squared loss is adopted, the loss function can be defined as $\ell(y, g) = \frac{1}{4}(yg(\mathbf{x}; \boldsymbol{\theta}) - 1)^2$.

A simple strategy to conduct multi-label classification is to prepare C binary classifiers. Here, we denote the binary classifier that determines whether $\mathbf{x}$ belongs to class t as $g_t(\mathbf{x}; \boldsymbol{\theta}_t)$.

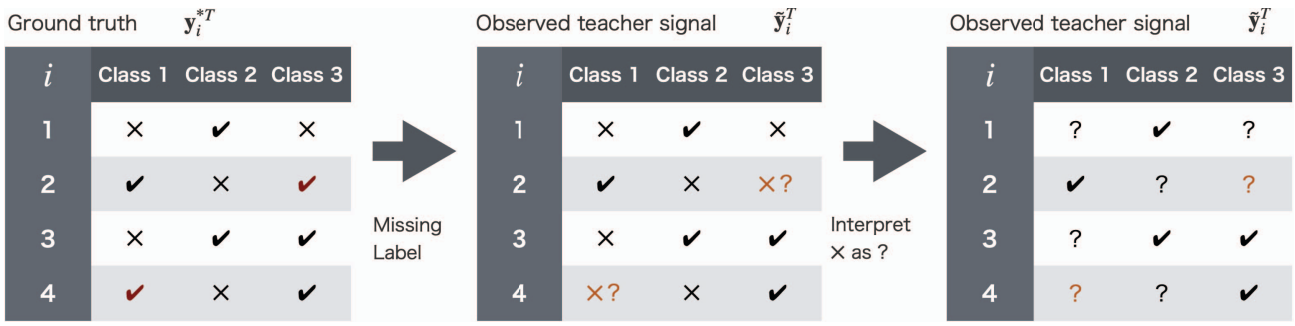
The expected value of function $f(\mathbf{x})$ according to the distribution of random variable $\mathbf{x}$ is expressed as $\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})}[f(\mathbf{x})]$. The class prior for class t is presented by π_t. $\pi_t = p([\mathbf{y}^*]_t = +1)$.

B. PU learning

First, we consider traditional positive and negative learning (PN learning) for class t. Here, the joint distribution of the sample and true teacher signal is denoted as $p(\mathbf{x}, [\mathbf{y}^*]_t)$. In traditional PN learning, learning is defined such that the PN risk represented by (1) is minimized.

$$R_{\mathrm{PN}}(\boldsymbol{\theta}_t) = \mathbb{E}_{\mathbf{x}, y \sim p(\mathbf{x}, [\mathbf{y}^*]_t)}[\ell\left(y, g_t(\mathbf{x}; \boldsymbol{\theta}_t)\right)] \tag{1}$$

Here, (1) can be written as (2) using the average of the positive and negative samples.



i	Class 1	Class 2	Class 3
1	×	✔	×
2	✔	×	✔
3	×	✔	✔
4	✔	×	✔

i	Class 1	Class 2	Class 3
1	×	✔	×
2	✔	×	×?
3	×	✔	✔
4	×?	×	✔

i	Class 1	Class 2	Class 3
1	?	✔	?
2	✔	?	?
3	?	✔	✔
4	?	?	✔

Fig. 1. A check mark indicates that the sample belongs to the corresponding class (i.e., a positive sample), and a cross mark indicates that the sample does not belong to the corresponding class (i.e., a negative sample). Here, we can interpret that some positive examples of the ground truth are missing in the observed teacher signal. In this case, it is impossible to interpret other than the positive samples in the observed teacher signal as negative samples. We treat this as being unlabeled. This is considered a learning problem when the teacher signal is both positive and unlabeled.

$$
\begin{aligned}
R_{\mathrm{PN}}(\boldsymbol{\theta}_t) =& \pi_t \mathbb{E}^{\mathrm{P}_t}[\ell(1, g_t(\mathbf{x}; \boldsymbol{\theta}_t))] \\
&+ (1-\pi_t)\mathbb{E}^{\mathrm{N}_t}[\ell(-1, g_t(\mathbf{x}; \boldsymbol{\theta}_t))] \\
\text{where, } \mathbb{E}^{\mathrm{P}_t}[\bullet] &= \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x} \mid [\mathbf{y}^*]_t=1)}[\bullet], \\
\mathbb{E}^{\mathrm{N}_t}[\bullet] &= \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x} \mid [\mathbf{y}^*]_t=-1)}[\bullet], \\
\pi_t &= p([\mathbf{y}^*]_t = 1).
\end{aligned} \tag{2}
$$

However, in PU learning, we cannot use (1) because $\mathbb{E}^{\mathrm{N}_t}[\bullet]$ in (2) cannot be approximated from the data. We should use the risk without the term $\mathbb{E}^{\mathrm{N}_t}[\bullet]$. Here, the idea is to transform the last term in (2) as follows [4]:

$$
\begin{aligned}
&(1-\pi_t)\mathbb{E}^{\mathrm{N}_t}[\ell(-1, g_t(\mathbf{x}; \boldsymbol{\theta}_t))] \\
&\quad = \mathbb{E}[\ell(-1, g_t(\mathbf{x}))] - \pi_t \mathbb{E}^{\mathrm{P}_t}[\ell(-1, g_t(\mathbf{x}; \boldsymbol{\theta}_t))]
\end{aligned}
$$

With this transformation, the PU risk can be written as follows:

$$
\begin{aligned}
R_{\mathrm{PU}}(\boldsymbol{\theta}_t) = \pi_t \mathbb{E}^{\mathrm{P}_t}&\left[\ell(1, g_t(\mathbf{x}; \boldsymbol{\theta}_t)) - \ell(-1, g_t(\mathbf{x}; \boldsymbol{\theta}_t))\right] \\
&+ \mathbb{E}\left[\ell(-1, g_t(\mathbf{x}; \boldsymbol{\theta}_t))\right]
\end{aligned} \tag{3}
$$

Note that (3) does not require an expectation value for negative examples; thus, we can use this risk for PU learning. By rewriting (3) in a form that can be approximated using the data, we get the following:

$$
\begin{aligned}
\widehat{R}_{\mathrm{PU}}(\boldsymbol{\theta}_t) = \frac{\pi_t}{n^{\mathrm{P}_t}} \sum_{i=1}^{n^{\mathrm{P}_t}} &\left[\ell\left(1, g_t(\mathbf{x}_i^{\mathrm{P}_t}; \boldsymbol{\theta}_t)\right)\right. \\
&\left. - \ell\left(-1, g_t(\mathbf{x}_i^{\mathrm{P}_t}; \boldsymbol{\theta}_t)\right)\right] \\
&+ \frac{1}{n}\sum_{i=1}^{n} \ell\left(-1, g_t(\mathbf{x}_i; \boldsymbol{\theta}_t)\right).
\end{aligned} \tag{4}
$$

Here, $\mathbf{x}_i^{\mathrm{P}_t}$ denotes cases that the i-th sample is labeled as class t. n^{P_t} is the number of samples labeled as class t, and n is the total number of all samples. If the "selected completely at random" (SCAR) assumption [16] holds, then $\mathbb{E}[\widehat{R}_{\mathrm{PU}}] = R_{\mathrm{PN}}$, and PU learning can be expected to yield the same classification performance as PN learning. The SCAR assumption can be presented mathematically as follows:

$$
p(\mathbf{x} \mid [\mathbf{y}^*]_t = +1) = p(\mathbf{x} \mid [\tilde{\mathbf{y}}]_t = +1). \tag{5}
$$

C. Estimation of π_t

Note that the class prior π_t is required to calculate $\widehat{R}_{\mathrm{PU}}(\boldsymbol{\theta}_t)$ in (4). Due to the missing labels, $\frac{\#\text{ samples labeled with class } t}{\#\text{ all samples}}$ does not imply π_t. Here, we explain how π_t is estimated using the method presented by du Plessis et al. [17].

Partial matching under the "not strongly overlapping" (NSO) assumption can be used to estimate π_t. The NSO assumption can be expressed as follows.

$$
p(\mathbf{x} \mid [\mathbf{y}^*]_t = +1) p(\mathbf{x} \mid [\mathbf{y}^*]_t = -1) \simeq 0 \text{ for } \mathbf{x} \in \mathcal{X}. \tag{6}
$$

Examples of the NSO assumption are shown in Figure 2.

Partial matching is the process of adjusting $\pi_t p(\mathbf{x} \mid [\mathbf{y}^*]_t = +1)$ to make it closer to $p(\mathbf{x})$. Here, we want to find π_t when the two distributions are closest to each other. The illustration of partial matching is shown in Figure 3.

Using Pearson divergence for the distance between distributions, π_t can be obtained by minimizing the following:

$$
\mathrm{PE}(\pi_t) = \frac{1}{2}\int \left(\frac{\pi_t p(\mathbf{x} \mid [\tilde{\mathbf{y}}]_t = +1)}{p(\mathbf{x})} - 1\right)^2 p(\mathbf{x}) d\mathbf{x}. \tag{7}
$$

du Plessis et al. [17] used the following equation to calculate (7) from the data:

$$
\widehat{\mathrm{PE}}(\pi_t) = \pi_t^2 \hat{\mathbf{h}}_t^T \hat{\mathbf{G}}^{-1} \hat{\mathbf{h}} - \pi_t^2 \frac{1}{2} \hat{\mathbf{h}}_t^T \hat{\mathbf{G}}^{-1} \hat{\mathbf{H}} \hat{\mathbf{G}}^{-1} \hat{\mathbf{h}}_t - \pi_t + \frac{1}{2}, \tag{8}
$$

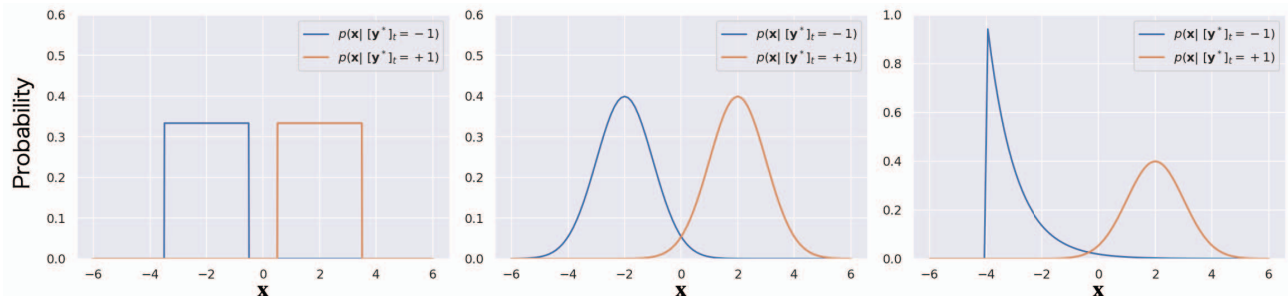


Fig. 2. Examples that follow the NSO assumption. The assumption is that the overlap between conditional distributions $p(\mathbf{x}|\ [\mathbf{y}^*]_t = -1)$ and $p(\mathbf{x}|\ [\mathbf{y}^*]_t = +1)$ for belonging to class t is small.

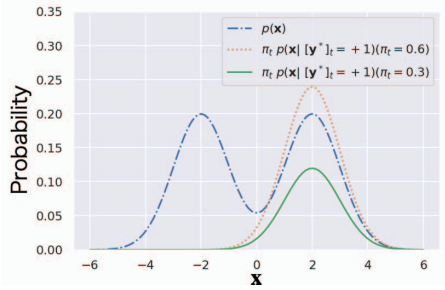


Fig. 3. Conceptual illustration of partial matching. Adjusting π_t changes the height of the distribution of positive samples. Partial matching means adjusting the height of the distribution of positive samples to match the distribution of all samples.

where

$$\hat{\mathbf{h}}_t = \frac{1}{n^{\mathrm{P}_t}} \sum_{i=1}^{n^{\mathrm{P}_t}} \boldsymbol{\varphi}(\mathbf{x}^{\mathrm{P}_t}),$$

$$\hat{\mathbf{H}} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\varphi}(\mathbf{x}) \boldsymbol{\varphi}(\mathbf{x})^T,$$

$$\hat{\mathbf{G}} = \hat{\mathbf{H}} + \lambda \mathbf{I}.$$

Here, $\boldsymbol{\varphi}(\mathbf{x})$ is a kernel function, and λ is a regularization parameter that should be specified with a small nonnegative number.

By minimizing (8) for π_t, we obtain the following:

$$\hat{\pi}_t = \left[2\hat{\mathbf{h}}_t^T \hat{\mathbf{G}}^{-1} \hat{\mathbf{h}} - \hat{\mathbf{h}}_t^T \hat{\mathbf{G}}^{-1} \hat{\mathbf{H}} \hat{\mathbf{G}}^{-1} \hat{\mathbf{h}}_t \right]^{-1}. \tag{9}$$

D. Algorithm

During training, we follow Algorithm 1, and for inference, we follow Algorithm 2. Conceptual diagrams of these algorithms are shown, where Algorithm 1 corresponds to Figure 4, and Algorithm 2 corresponds to Figure 5.

Algorithm 1: Multi-label learning with incomplete-label dataset by one-vs-rest strategy

input : Observed data (train data) $(\mathbf{x}_i, \tilde{\mathbf{y}}_i)\ (i = 1, ..., n)$,
Binary classifier $g(\mathbf{x}; \boldsymbol{\theta})$,
Loss function for positive and negative learning $\ell(y, g(\mathbf{x}; \boldsymbol{\theta}))$

output: Trained classifier $g_t(\mathbf{x}, \hat{\boldsymbol{\theta}}_t)$ for each class $(t = 1, 2, ..., C)$

for t in $1..C$ **do**

$\mathbf{X}^{\mathrm{P}_t} \leftarrow \{\mathbf{x}_i | [\tilde{\mathbf{y}}_i]_t = 1\}$

$n^{\mathrm{P}_t} \leftarrow |\mathbf{X}^{\mathrm{P}_t}|$

$\{\mathbf{x}_1^{\mathrm{P}_t}, ..., \mathbf{x}_{n^{\mathrm{P}_t}}^{\mathrm{P}_t}\} \leftarrow \mathbf{X}^{\mathrm{P}_t}$ // reassign indices

$\hat{\pi}_t \leftarrow \left[2\hat{\mathbf{h}}_t^T \hat{\mathbf{G}}^{-1} \hat{\mathbf{h}} - \hat{\mathbf{h}}_t^T \hat{\mathbf{G}}^{-1} \hat{\mathbf{H}} \hat{\mathbf{G}}^{-1} \hat{\mathbf{h}}_t \right]^{-1}$

$$\widehat{R}_{\mathrm{PU}}(\boldsymbol{\theta}_t) := \frac{\hat{\pi}_t}{n^{\mathrm{P}_t}} \sum_{i=1}^{n^{\mathrm{P}_t}} \left[\ell\left(1, g_t(\mathbf{x}_i^{\mathrm{P}_t}; \boldsymbol{\theta}_t)\right) - \ell\left(-1, g_t(\mathbf{x}_i^{\mathrm{P}_t}; \boldsymbol{\theta}_t)\right) \right] + \frac{1}{n} \sum_{i=1}^{n} \ell\left(-1, g_t(\mathbf{x}_i; \boldsymbol{\theta}_t)\right)$$

$\hat{\boldsymbol{\theta}}_t \leftarrow \arg\min_{\boldsymbol{\theta}_t} \widehat{R}_{\mathrm{PU}}(\boldsymbol{\theta}_t)$

end

Algorithm 2: Multi-label classification by one-vs-rest strategy

input : Data to predict $\mathbf{x}_i\ (i = 1, ..., n)$,
Trained Binary classifier $g_t(\mathbf{x}; \hat{\boldsymbol{\theta}}_t)$,

output: Predicted labels $\hat{\mathbf{y}}_i\ (i = 1, ..., n)$

for i in $1..n$ **do**

for t in $1..C$ **do**

$s_{i,t} \leftarrow g_t(\mathbf{x}_i; \hat{\boldsymbol{\theta}}_t)$

$$[\hat{\mathbf{y}}_i]_t \leftarrow \begin{cases} +1 & (s_{i,t} > 0) \\ -1 & (s_{i,t} \leq 0) \end{cases}$$

end

end

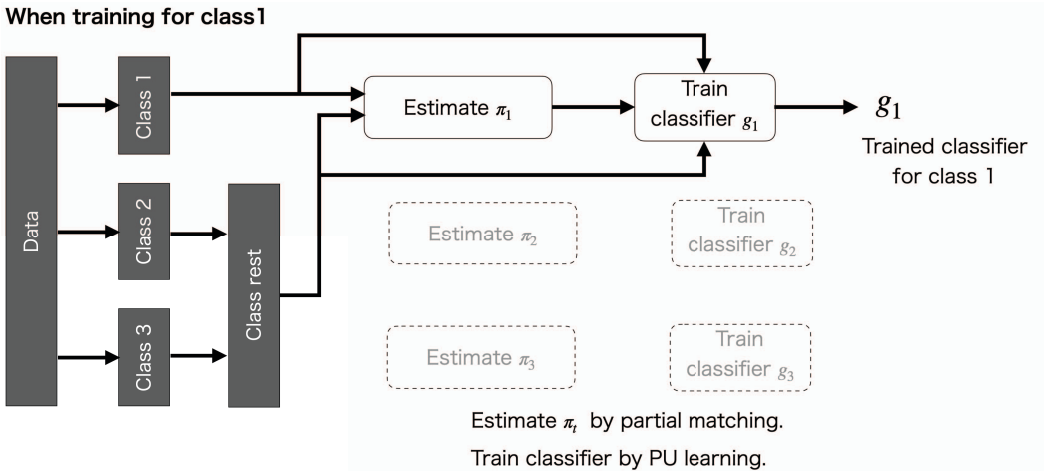


Fig. 4. Conceptual diagram for Algorithm1

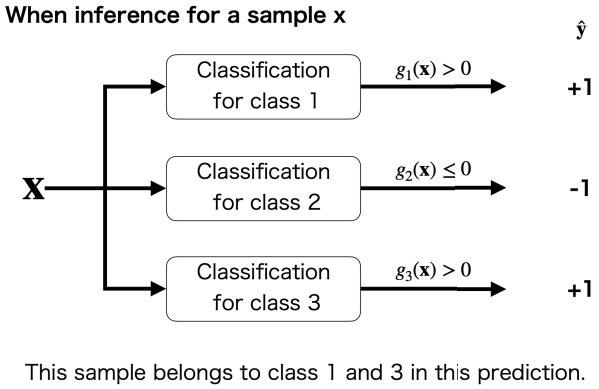


Fig. 5. Conceptual diagram for Algorithm2

IV. Experiments

A. Experimental Dataset

We created the experimental dataset using information collected from the NVD on May 5th, 2020. The basic information of the dataset is described in Tabel I.

TABLE I
DATASET USED IN THE EXPERIMENT.

Number of samples	93,751
Number of features	2,836
Feature representation	bag of words (binary)
Number of classes	49 (occurence $\geq$ 200)

B. Evaluation

In these experiments, we compared the traditional PNmulti machine learning method and the proposed PUmulti method). In both cases, the binary classifier was a linear classifier $g(\mathbf{x}; \boldsymbol{w}) = \boldsymbol{w}^T\mathbf{x} + w_0$, and the loss function was the squared error $\ell(y, g) = \frac{1}{4}(yg(\mathbf{x}; \boldsymbol{\theta}) - 1)^2$. The ground truth (true teacher signal $\mathbf{y}^*$), which was required for these evaluations, was created by sampling 1000 CVE entries randomly and labeling them by experts. The experimental procedure is shown in Figure 6.

PUmulti is used to denote the proposed method. Here, Algorithm 1 was used to learn the classifier. The kernel required for estimating π_t was the identity map $\varphi(\mathbf{x}) = \mathbf{x}$.

PNmulti is used to denote the method which does not consider missing labels. Here, when learning a classifier, missing label ($\tilde{\mathbf{y}}_t = 0$) was interpreted as a negative example (-1), and the classifier was trained to minimize the PN loss in (1).

In the following, we explain the metrics used to evaluate multi-label classification, i.e., the mean F1 score and mean Jaccard index.

1) Mean F1 score: The mean F1 score is the average of the F1 scores for each sample. Here, if the F1 score for the i-th test sample is denoted as $\mathrm{F1}_i$, the mean F1 score is calculated as follows,

$$\text{Mean F1 score} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{F1}_i. \tag{10}$$

The F1 score for the i-th sample is calculated as follows,

$$\mathrm{F1}_i = \frac{2}{\frac{1}{\mathrm{Pre}_i} + \frac{1}{\mathrm{Rec}_i}}. \tag{11}$$

where Pre_i and Rec_i are calculated as follows,

$$\mathrm{Pre}_i = \frac{\hat{\mathbf{y}}_i'^T\mathbf{y}_i^{*\prime}}{|\mathbf{y}_i^{*\prime}|_{L0}},\ \mathrm{Rec}_i = \frac{\hat{\mathbf{y}}_i'^T\mathbf{y}_i^{*\prime}}{|\hat{\mathbf{y}}_i'|_{L0}}. \tag{12}$$

Here, $|\bullet|_{L0}$ represents the number of nonzero elements in the vector, and "$\prime$" means replacing -1 elements with 0 ($\hat{\mathbf{y}}_i' = \frac{\hat{\mathbf{y}}_i+1}{2}$, $\mathbf{y}_i^{*\prime} = \frac{\mathbf{y}_i^*+1}{2}$).

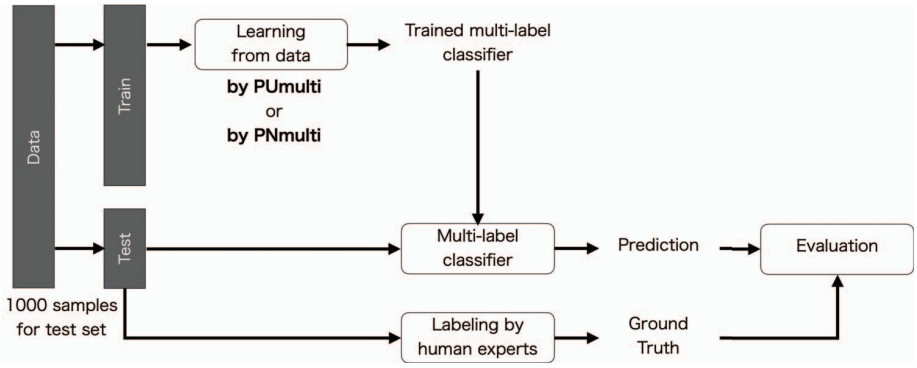


Fig. 6. Experimental procedures for PUmulti and PNmulti. 1000 CVE entries were selected randomly and labeled by experts.

TABLE II
MULTI-LABEL CLASSIFICATION PERFORMANCE

Metrics	PUmulti	PNmulti
Mean F1 score	**0.5066**	0.4044
Mean Jaccard index	**0.4033**	0.3348

2) Mean Jaccard index: The Jaccard index is a measure of similarity of two sets. The mean Jaccard index is the average of the Jaccard index for prediction $\hat{\mathbf{y}}_i$ and the true label $\mathbf{y}_i^*$ calculated on each sample,

$$\text{Mean Jaccard index} = \frac{1}{n}\sum_{i=1}^{n}\frac{\hat{\mathbf{y}}_i'^T\mathbf{y}_i^{*\prime}}{|\hat{\mathbf{y}}_i' + \mathbf{y}_i^{*\prime}|_{L0}}. \quad (13)$$

3) F1 score: The F1 score for class t is expressed as follows,

$$\text{F1 score}_t = \frac{2}{\frac{1}{\text{Pre}_t} + \frac{1}{\text{Rec}_t}}. \quad (14)$$

Here, Pre_t and Rec_t denote the precision and recall for class t,

$$\text{Pre}_t = \frac{[\hat{Y}']_{:,t}^T[Y^{*\prime}]_{:,t}}{|[Y^{*\prime}]_{:,t}|_{L0}},\ \text{Rec}_t = \frac{[\hat{Y}']_{:,t}^T[Y^{*\prime}]_{:,t}}{|[\hat{Y}']_{:,t}|_{L0}}, \quad (15)$$

where, $\hat{Y}' = (\hat{\mathbf{y}}_1', \hat{\mathbf{y}}_2', ..., \hat{\mathbf{y}}_n')^T$, $Y^{*\prime} = (\mathbf{y}_1^{*\prime}, \mathbf{y}_2^{*\prime}, ..., \mathbf{y}_n^{*\prime})^T$. $[Y]_{:,t}$ is the vector of the t-th columns in Y.

V. RESULTS

The evaluation results obtained using 1000 cases labeled by experts are shown in Table II. As can be seen, the classification performance of the proposed method was better than that of PNmulti, which does not consider missing labels.

The performance for the top-10 frequently occurring classes is shown in Table III. As shown, PUmulti outperformed PNmulti in many classes. Specifically, PNmulti outperformed PUmulti in only three of 49 classes.

The reason PUmulti outperformed PNmulti can be demonstrated by the distribution of the number of labels per sample. As shown in Figure 7, approximately 34% of the samples in PNmulti have no labels at all. In contrast, with PUmulti, only 2% of the samples are labeled, i.e., nearly all samples have some kind of label. Therefore, the fact that approximately 34% of the samples are unlabeled suggests that PNmulti is learning incorrectly. In case of missing labels, the unlabeled samples should be interpreted as "unlabeled" and trained. However, PNmulti interprets unlabeled samples as "negative," which results in the assignment of only a few labels.

TABLE III
BINARY CLASSIFICATION PERFORMANCE FOR EACH CLASS

CWE-ID	**PUmulti**	PNmulti	# samples
CWE-74	**0.1169**	0.0000	315
CWE-20	**0.2929**	0.1399	305
CWE-284	**0.0829**	0.0000	200
CWE-119	0.8541	**0.8727**	141
CWE-79	**0.9745**	0.9743	139
CWE-269	**0.3888**	0.0253	78
CWE-287	**0.3168**	0.1860	76
CWE-200	**0.7558**	0.7086	73
CWE-89	**0.9784**	**0.9784**	68
CWE-94	**0.3255**	0.1739	62
CWE-125	**0.7441**	0.7222	44

As shown in Figure 7, PUmulti tends to label less than ground truth, which suggests that the data do not follow the SCAR assumption, which is necessary for PU learning to work as the theory suggests (the SCAR assumption assumes that missing labels occur uniformly at random). However, this is not the case in reality. For example, there may be a tendency to assign CWE-IDs that are perceived to be more threatening. In such cases, the SCAR assumption for the data is invalid. In fact, 99% of CVE entries have only a single CWE-ID. If labels are missing at random, there would be many more CVE entries with no CWE-ID and many more CVE entries with multiple CWE-IDs. Although the data may not follow these assumptions, the results are better for PUmulti than PNmulti, as shown in Table II and Table III.

VI. CASE STUDY

In this section we discuss a series of concrete examples to demonstrate the validity of the inferred labels obtained by the proposed method.

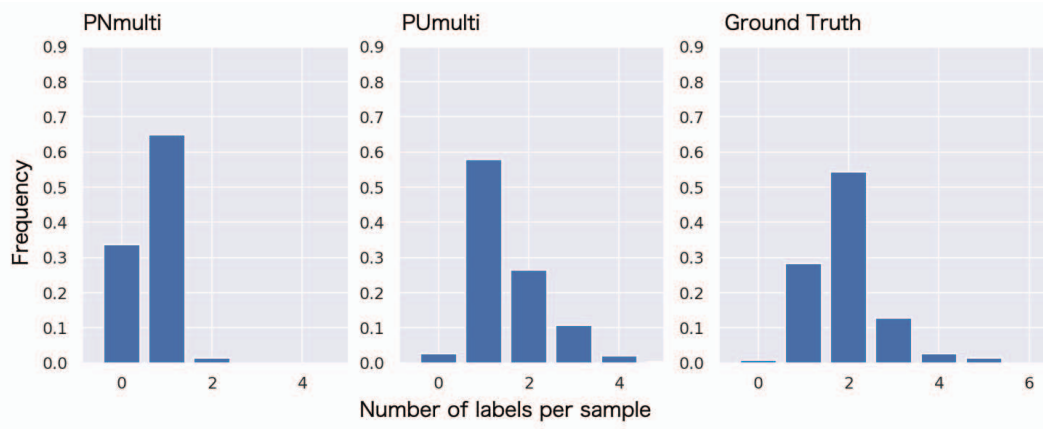


Fig. 7. Distribution of the number of labels per sample in inference. PNmulti did not assign labels to approximately 34% of the samples, which suggests that PNmulti's learning is incorrect.

CVE-2006-1552

Integer overflow in ImageIO in Apple Mac OS X 10.4 up to 10.4.5 allows remote attackers to cause a denial of service (crash) via a crafted JPEG image with malformed JPEG metadata, as demonstrated using Safari, aka "Deja-Doom". [18]

This CVE entry was originally labeled as CWE-189 (CWE CATEGORY: numeric errors). In the inference of the proposed method, in addition to CWE-189, CWE-190 (Integer Overflow or Wraparound) was given. CWE-190 has more detailed description about integer overflow.

CVE-2008-0114

Unspecified vulnerability in Microsoft Excel 2000 SP3 through 2003 SP2, Viewer 2003, and Office for Mac 2004 allows user-assisted remote attackers to **execute arbitrary code** via crafted Style records that **trigger memory corruption**. [19]

This CVE entry was originally labeled as CWE-94 (improper control of generation of code ('code injection')). In the inference of the proposed method, CWE-119 (improper restriction of operations within the bounds of a memory buffer) was given in addition to CWE-94. Here, memory corruption is mentioned at the end of the sentence; thus, it is reasonable that CWE-119 is also given.

CVE-2008-3550

The CQWeb login page in IBM Rational ClearQuest 7.0.1 allows remote attackers to **obtain potentially sensitive information** (page source code) via a combination of ?script? and ?/script? sequences in the id field, **possibly related to a cross-site scripting (XSS) vulnerability.** [3]

This CVE entry was originally labeled as CWE-200 (exposure of sensitive information to an unauthorized actor). In addition to CWE-200, in the inference of the proposed method, the label CWE-79 (improper neutralization of input during web page generation ('cross-site scripting')) was assigned. It is mentioned as "possibly related to a cross-site scripting" in the description.

A chord diagram to check the co-occurrence relations when multiple CWE-IDs were inferred is shown in Fig. 8, where the thickness of the line connecting the two CWE-IDs indicates the number of times they appeared simultaneously.

Figure 9 shows examples of CWE-IDs that co-occur frequently. In the left example, CWE-190 often co-occurs with CWE-189. If a CVE entry belongs to "integer overflow," it also belongs to "numeric errors," so the result is satisfactory. A concrete example is CVE-2018-19107.

CVE-2018-19107

In Exiv2 0.26, Exiv2::IptcParser::decode in iptc.cpp (called from psdimage.cpp in the PSD image reader) may suffer from a denial of service (heap-based **buffer over-read**) caused by an **integer overflow** via a crafted PSD image file. [20]

This CVE entry was originally labeled as CWE-190. With the proposed method, in addition to CWE-190, the labels CWE-189 and CWE-125 (out-of-bounds read) were assigned. In addition to the relation that if CWE-190, then CWE189, CWE-125 was also successfully assigned based on "buffer over-read" in the sentence.

As shown on the right side of Figure 9, CWE-74 also co-occurs with CWE-77 and CWE-78. CWE-74, CWE-77, and CWE-78 are named "injection," "command Injection," and "OS command injection," respectively, and have the common feature of injection. Therefore, it is reasonable for them to co-occur. A concrete example is CVE-2018-16660.

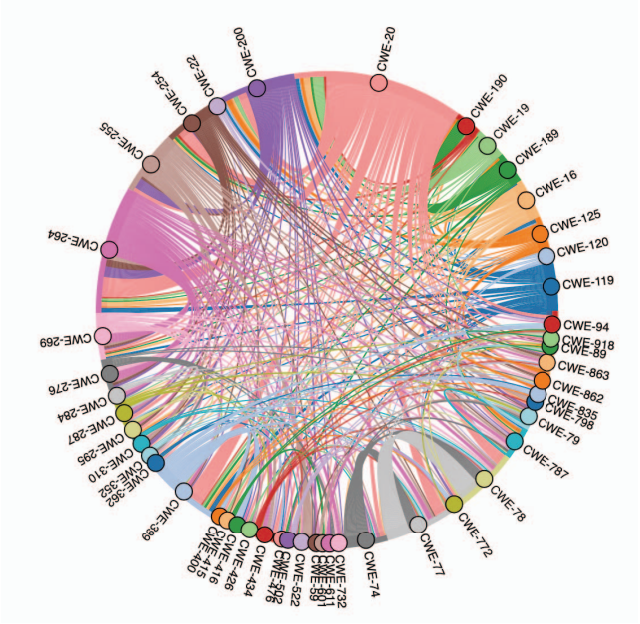


Fig. 8. Visualization of co-occurrence relations of CWE-IDs with labels inferred by our proposed method. The thickness of the line connecting two CWE-IDs indicates the number of times they co-occurred. The line color has no specific meaning.

CVE-2018-16660

A **command injection vulnerability** in PWS in Imperva SecureSphere 13.0.0.10 and 13.1.0.10 Gateway allows an attacker with authenticated access to **execute arbitrary OS commands** on a vulnerable installation. [21]

This CVE entry was originally labeled as CWE-78. In the inference of the proposed method, CWE-74 and CWE-77 were assigned to it in addition to CWE-78.

As shown in Figure 10, both CWE-20 and CWE-264 can be associated with a variety of CWE-IDs. CWE-20 represents "improper input validation," which is a very abstract concept. CWE-264 is also an abstract concept: "permissions, privileges, and access controls." Since inadequate privileges can lead to various vulnerabilities, it is also reasonable to associate CWE-264 with various CWE-IDs.

VII. Conclusion

We have observed that many CVE entries stored in NVD are assigned with incomplete CWE-ID information. In addition, most CVE entries have a single CWE-ID, even though they fit into multiple CWE-IDs. In this paper, we have proposed a system that uses machine learning to assign multiple related CWE-IDs to CVE entries. We interpreted this problem as a multi-label classification with missing labels, and we adapted PU learning for each class. The experimental results demonstrate that the proposed method outperforms the traditional method, which does not consider missing labels. In addition, in case studies, we confirmed that multiple CWE-IDs assigned by the proposed method provide compensate information to pre-existing CWE-IDs.

References

[1] "NVD - Home," https://nvd.nist.gov/, accessed: Apr. 6, 2021.

[2] "CWE - Common Weakness Enumeration," https://cwe.mitre.org/, accessed: Apr. 6, 2021.

[3] "NVD - CVE-2008-3550," https://nvd.nist.gov/vuln/detail/CVE-2008-3550, accessed: Apr. 6, 2021.

[4] M. C. du Plessis, G. Niu, and M. Sugiyama, "Analysis of learning from positive and unlabeled data," in *Advances in Neural Information Processing Systems*, 2014, pp. 703–711.

[5] M. Bozorgi, 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*, 2010, pp. 105–114.

[6] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng, "Learning to predict severity of software vulnerability using only vulnerability description," in *IEEE International Conference on Software Maintenance and Evolution*, 2017, pp. 125–136.

[7] Z. Han, X. Li, H. Liu, Z. Xing, and Z. Feng, "Deepweak: Reasoning common software weaknesses via knowledge graph embedding," in *25th International Conference on Software Analysis, Evolution and Reengineering*, 2018, pp. 456–466.

[8] S. Nakagawa, T. Nagai, H. Kanehara, K. Furumoto, M. Takita, Y. Shiraishi, T. Takahashi, M. Mohri, Y. Takano, and M. Morii, "Character-level convolutional neural network for predicting severity of software vulnerability from vulnerability description," *IEICE Transactions on Information and Systems*, vol. E102.D, pp. 1679–1682, 2019.

[9] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai, "Have things changed now?: an empirical study of bug characteristics in modern open source software," in *Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability*, 2006, pp. 25–33.

[10] S. Na, T. Kim, and H. Kim, "A study on the classification of common vulnerabilities and exposures using naïve bayes," in *Advances on Broad-Band Wireless Computing, Communication and Applications*, 2016, pp. 657–662.

[11] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, and T. Takahash, "Automation of vulnerability classification from its description using machine learning," in *IEEE Symposium on Computers and Communications*. IEEE, 2020, pp. 1–7.

[12] E. Aghaei, W. Shadid, and E. Al-Shaer, "Threatzoom: CVE2CWE using hierarchical neural network," *CoRR*, vol. abs/2009.11501, 2020.

[13] S. S. Bucak, R. Jin, and A. K. Jain, "Multi-label learning with incomplete class assignments," in *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2011, pp. 2801–2808.

[14] A. Kanehira and T. Harada, "Multi-label ranking from positive and unlabeled data," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016, pp. 5138–5146.

[15] X. Kong, Z. Wu, L. Li, R. Zhang, P. S. Yu, H. Wu, and W. Fan, "Large-scale multi-label learning with incomplete label assignments," in *Proceedings of the 2014 SIAM International Conference on Data Mining*. SIAM, 2014, pp. 920–928.

[16] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 213–220.

[17] M. C. du Plessis and M. Sugiyama, "Class prior estimation from positive and unlabeled data," *IEICE Trans. Inf. Syst.*, vol. 97-D, no. 5, pp. 1358–1362, 2014.

[18] "NVD - CVE-2006-1552," https://nvd.nist.gov/vuln/detail/CVE-2006-1552, accessed: Apr. 6, 2021.

[19] "NVD - CVE-2008-0114," https://nvd.nist.gov/vuln/detail/CVE-2008-0114, accessed: Apr. 6, 2021.

[20] "NVD - CVE-2018-19107," https://nvd.nist.gov/vuln/detail/CVE-2018-19107, accessed: Apr. 6, 2021.

[21] "NVD - CVE-2018-16660," https://nvd.nist.gov/vuln/detail/CVE-2018-16660, accessed: Apr. 6, 2021.

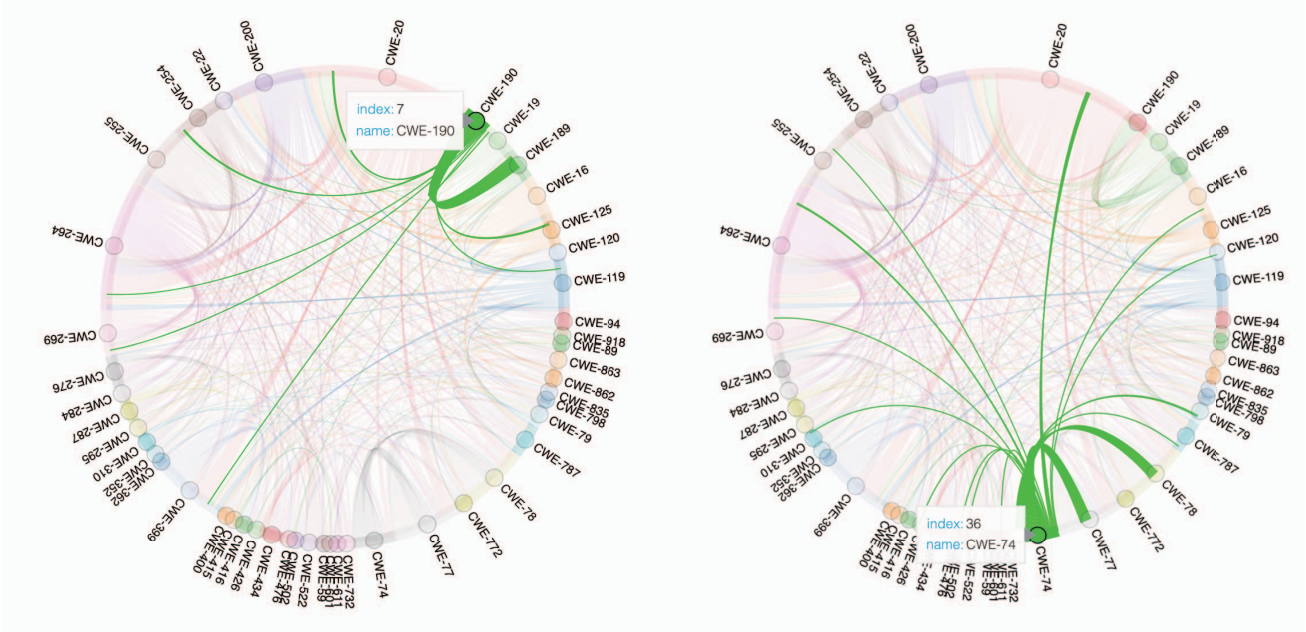


Fig. 9. The left side shows that CWE-190 (Integer Overflow or Wraparound) often appears with CWE-189 (Numeric Errors). The right side shows that CWE-74 (Improper Neutralization of Special Elements in Output Used by a Downstream Component ("Injection")) often appears with CWE-77 (Improper Neutralization of Special Elements used in a Command ("Command Injection")) and CWE-78 (Improper Neutralization of Special Elements used in an OS Command ("OS Command Injection")).

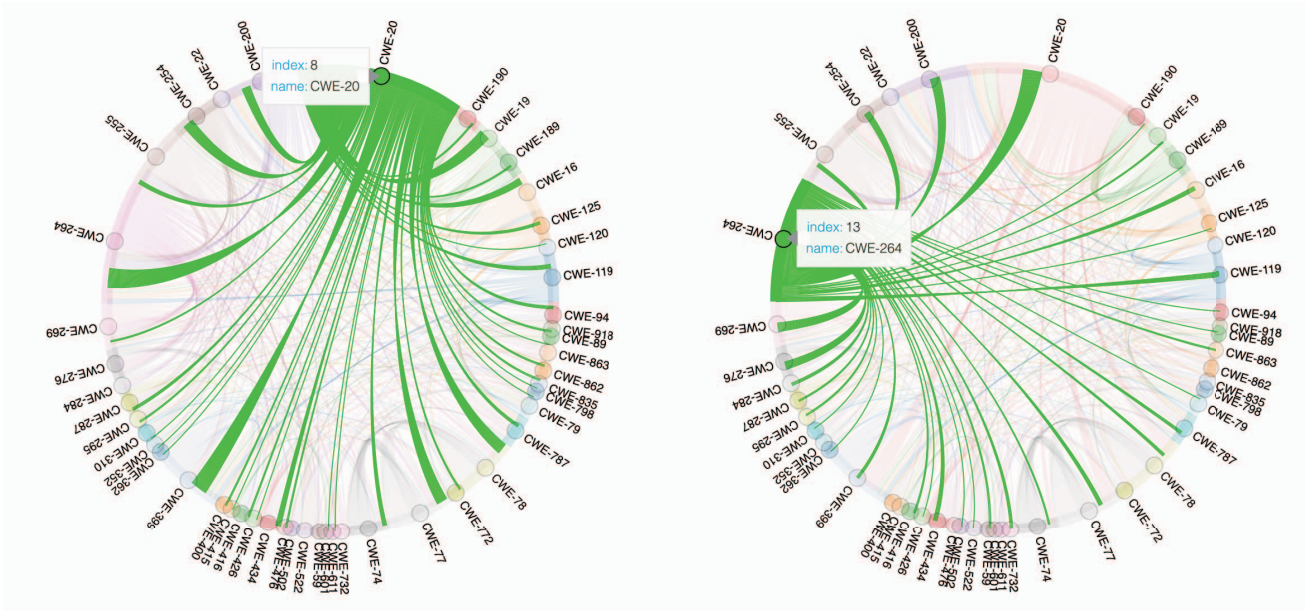


Fig. 10. CWE-20 (left) and CWE-264 (right) are abstract CWE-IDs related to many other CWE-IDs. The inference of the proposed method captured this relationship.