An Approach to Discover and Assess Vulnerability Severity Automatically in Cyber-Physical Systems

Yuning Jiang
University of Skövde
Skövde, Sweden
yuning.jiang@his.se

Yacine Atif
University of Skövde
Skövde, Sweden
yacine.atif@his.se

ABSTRACT

Current vulnerability scoring mechanisms in complex cyber-physical systems (CPSs) face challenges induced by the proliferation of both component versions and recurring scoring-mechanism versions. Different data-repository sources like National Vulnerability Database (NVD), vendor websites as well as third party security tool analysers (e.g. ICS CERT and VulDB) may provide conflicting severity scores. We propose a machine-learning pipeline mechanism to compute vulnerability severity scores automatically. This method also discovers score correlations from established sources to infer and enhance the severity consistency of reported vulnerabilities. To evaluate our approach, we show through a CPS-based case study how our proposed scoring system automatically synthesises accurate scores for some vulnerability instances, to support remediation decision-making processes. In this case study, we also analyse the characteristics of CPS vulnerability instances.

ACM Reference Format:
Yuning Jiang and Yacine Atif. 2020. An Approach to Discover and Assess Vulnerability Severity Automatically in Cyber-Physical Systems. In *13th International Conference on Security of Information and Networks (SIN 2020), November 4–7, 2020, Merkez, Turkey.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3433174.3433612

1 INTRODUCTION

The rapid advancement of information and communication technologies enables seamless integration of new software and hardware, to shift from human-controlled mechanisms to autonomous mechanisms empowered by cyber-physical systems (CPS) that are currently driving Industry 4.0 evolution. However, vulnerabilities emerge intermittently in different software, hardware, and firmware. These undesirable instances of vulnerability need to be rooted out from CPS infrastructure and their severity needs to be assessed and ranked, to prioritise patches that prevent threat-induced anomalies or intrusion attempts [7].

Modern security practices promote quantitative analysis-methods to provide predictive analytics and support patching prioritisation exercises. A typical approach uses the standard Common Vulnerability Scoring System (CVSS) for vulnerability rating, which requires manual inputs from an expert analyst to evaluate a vulnerability instance following some given metrics [13][9]. One primary data source is the Common Vulnerabilities and Exposures (CVE) database, the vulnerability reports from which are further assigned with CVSS scores and other statistical views by National Vulnerability Database (NVD). Yet, making vulnerability-remediation decisions based solely on CVE or NVD records can be biased and misleading [4]. A knowledge-base that consolidates multiple sources through sound classification rules appears to be missing, yet it promises a suitable level of decision-support. While CVSS mechanism is commonly used [13][9], inconsistent scores exist for the same vulnerability instance, as vulnerability properties change over time and across different deployment environments. Other relevant data sources, such as vendor reports, reviews from third parties, and online security forums or blogs, can also be leveraged to factor-in weights over existing CVSS scores [8].

In this paper, a vulnerability-severity scoring system is proposed to reflect the severity of a vulnerability instance. Our proposed scoring system enhances compatibility across different CVSS versions, while streamlining vulnerability analysis. In doing so, we adopt standard CVSS metrics and related scoring-mechanism to quantify the difficulty to exploit vulnerabilities and the magnitude of resulting impacts from using those exploits maliciously. We also correlate existing CVSS scores of vulnerability instances provided by different online cybersecurity data sources, such as NVD, vendor websites or technical reports from third party reviews (e.g. ICS CERT and VulDB), to consolidate scores in a way that better describe the actual severity of vulnerability instances. Our proposed method automates vulnerability assessment and related parameters' evaluation for CPS based infrastructures [12].

The main contributions of this work can be summarised as follows:

- A novel machine-learning based pipeline that streamlines the process of vulnerability assessment and computes CVSS severity scores for vulnerability instances. This proposed methods considers and reconciles inconsistent scores using majority voting, to improve the quality of training ground for the machine-learning models. Our model also allows easy customisation on selecting a preferable CVSS version, to allow vulnerability assessment while prioritising a consistent and common semantic.
- A case study centred around CPS vulnerability analysis to validate the proposed machine-learning based vulnerability-assessment approach. In this case study, an approach to design a proper query logic for vulnerability-instance retrievals is also introduced.

SIN 2020, November 4–7, 2020, Merkez, Turkey

ACM ISBN 978-1-4503-8751-4/20/09…$15.00
https://doi.org/10.1145/3433174.3433612

The rest of this paper is organised as follows: In Section II, we provide some background and formally state the problem addressed in this paper, followed by Section III which discusses standard vulnerability-severity metrics and related vulnerability-assessment processes used in the CVSS mechanism. In Section IV, we reveal our vulnerability assessment prototype, which correlates existing CVSS scores against other security-alert indicators, as well as reconciles different CVSS versions using some text-mining techniques on a corpus of vulnerability reports. In Section V, we evaluate our vulnerability-discovery and assessment methodology in CPS contexts through some analysis. In Section VI, we provide some concluding remarks and discuss some future research directions.

2 BACKGROUND AND PROBLEMS

The continuous evolution of risks in enterprises and the limited availability of cyber-attack data bring notable quantitative methods and potential metrics into security-related research agendas. This effort aims to enhance previous risk management frameworks that are mostly based on expert opinions. Among some initial attempts made towards combining emerging metrics, the standard mechanism CVSS is widely used to support vulnerability-severity assessment in both academic research and particularly in security-critical industrial domains [5][9][8][14]. However, some characteristics of CVSS mechanism are challenging in practice. More specifically, current CVSS-scoring process is mostly subjective, and incurs time delays to rate the severity of a vulnerability since its disclosure in CVE, which widens the risk window for potential zero-day attacks to occur [12]. Therefore, an automated vulnerability-score inference system can decrease this risk window. However, several limitations related to focus dilemmas require further investigation, like how to derive related measurements to determine these metric scales for a given vulnerability. On the other hand, compatibility issues among existing CVSS versions which result in different measurements of standard metrics was overlooked by previous research. Reported vulnerability instances are scored using different CVSS versions and adopted by various organisations [13], resulting in compatibility gaps. For instance, NVD provides CVSS version3 scores only to vulnerability instances reported in and after the year of 2015. In this paper we contribute a solution to overcome these gaps by providing a set of scores under different CVSS versions, while allowing users to select the preferable version.

Correlation studies between multiple cybersecurity data-sources have been undertaken to combine different perspectives, in order to connect multifaceted analysis into broader statistical associations. CVE, NVD and CERT are widely used vulnerability-analytics databases for uniquely identified vulnerability recordings. These databases are further correlated to data sources like ExploitDB. For instance, one study from Allodi and Massacci [1] correlates the NVD to ExploitDB and (Symantec) AttackSignature and ThreatExplorer. Another study from Geer and Roytman [6] also correlates the NVD database to ExploitDB and (Metasploit) to support penetration testers. A variety of approaches apply text-mining techniques not in cybersecurity reports but industrial blogs like Twitter [3] and security papers. For example, Zhu and Dumitras apply NLP to extract malware detection features from research papers automatically [18]. These works highlight that statistical interpretations of CVE/NVD datasets need to be combined with other live security-related data-sources, such as product vendors across deployed infrastructures, to raise indicators' reliability and precision. However, these works are usually in the field of software vulnerability analysis. Correlation studies considering different terminology used in cybersecurity and CPS domains are limited. In our method, we extracted the entities of vulnerable components and relevant vendor names in CVE vulnerability reports, and then map retrieved entities with the Common Platform Enumeration (CPE) as well as vendor websites, to generate dictionary for CPS components and vendors.

The retrieved information from cybersecurity data-sources support further pattern recognition and trend analysis. Using Artificial Intelligence techniques, large amounts of such open-source vulnerability data [14] can be analyzed. More specifically, machine-learning techniques like text-mining are applied to classify disclosed vulnerabilities automatically and to guide predictive analytics of security gaps, as suggested by Gawron et al., [5] and by Yamamoto et al., [16] in related previous studies. Nevertheless, using correlated cybersecurity data sources also raise potential inconsistencies, such as the disparity between scores of the same vulnerability instances [8]. One drawback of previous AI-based CVSS computing approaches is that they directly adopt the vulnerability reports from CVE and CVSS scores from NVD as training grounds, which may induce bias to their model. Instead, we correlate vulnerability instances in NVD to corresponding vendor reports and third-party cybersecurity analysers such as CERT reports, and integrate these information into a unified package as our training grounds. To be more specific, we use the vulnerability descriptions in NVD as training input. We then apply majority voting on inconsistence scores before using them as training grounds. In doing so, our approach streamlines the computation of vulnerability severity to address such inconsistencies, in order to optimise security resources allocation and to shorten the potential risk window.

3 VULNERABILITY SEVERITY METRICS

CVSS is developed and maintained by the CVSS Special Interest Group (CVSS-SIG) and reported in the Forum for Incident Response and Security Teams (FIRST). The development[1] of CVSS has gone through three amendment stages, and is currently in the stage of CVSS version 3 (V3). Using CVSS, a vulnerability instance could be identified by different types of properties measured through inherent metrics and following a range of corresponding measurement values [10]. The principal properties of an asset vulnerability are categorised into three groups, namely *Base*, *Temporal*, and *Environmental* properties. In this paper, we focus on the base property only. The base-property refers to inherent characteristics of a vulnerability that do not change over time or over deployment environments, which is the focus of this paper. Under CVSS V3, the base-property is further categorised into a)exploitability, b)scope and c)impact properties [9].

- Exploitability: The exploitability property is based on the possibility, difficulty and complexity of exploiting a vulnerable component. The exploitability of a vulnerability could be represented using four metrics, namely *AttackVector (AV)*, *AttackComplexity (AC)*, *PrivilegesRequired (PR)* as well as

[1] https://www.first.org/cvss/user-guide

UserInteraction (UI). Attack-path refers to the path through which the vulnerable component could be targeted and then exploited. Different attack-path values express the difficulty associated with exploitability. Attack-complexity describes the difficulty to make use of the vulnerability. The required privileges is a metric of the exploitability property that assesses the level of authority that a threat-agent needs to obtain to exploit the vulnerable asset. Finally User-interaction metric of exploitability property differs by the needed level of user's participation so that the vulnerable component could be compromised.

- Scope: The scope property of a vulnerability indicates whether exploiting that vulnerability could impact the vulnerability of other components. The metric associated with this property is called *ScopeChange* (S) and states whether exploiting that vulnerability on a component's scope gives access to other components' scope or not.
- Impact: The impact property of a vulnerability measures the consequences resulting from exploiting the vulnerability, which could cause losses in *Confidentiality* (C), *Integrity* (I) or *Availability* (A), i.e. (CIA) triad, of the corresponding component. The values used by CVSS to measure the severity of the impact metric on the vulnerable component are none- (N), low- (L) or high- (H) impact.

Ultimately, a vulnerability quantification system needs to deliver a CVSS vulnerability score based on some mathematical formulation. Equation 1 maps measurements of *Exploitability, Scope and Impact* property metrics to the *Base* category score of a vulnerability. Measurements for each property metric are collected with respect to component c_i in CPS components set C, as illustrated by Equation 2. Going bottom-up from measurements to score, each component c_i is subject to vulnerability v measured by its Base property-metric measurements: $Y^v_{Exploit}$, Y^v_{Scope} and Y^v_{Impact}. The function f^v_{Base} of Equation 1 maps these obtained measurements into a Base category score.

$$f^v_{Base} : P^v_{Exploit} * P^v_{Scope} * P^v_{Impact} \rightarrow P^v_{Base} \quad (1)$$

where vulnerability measurements vector v_i are collected for component c_i by:

$$f^v_{Exploit} : C \rightarrow P^v_{Exploit}, \text{Such as: } v_i^{Exploit} \mapsto f^v_{Exploit}(c_i) \quad (2)$$

$$f^v_{Impact} : C \rightarrow P^v_{Impact}, \text{Such as: } v_i^{Impact} \mapsto f^v_{Impact}(c_i) \quad (3)$$

4 DISCOVERING VULNERABILITY SEVERITY

In this section, we present a machine-learning (ML) based method to discover CVSS scores of reported vulnerability instances for which no score is provided. As illustrated in Fig. 1, we first adopt majority voting techniques [15] to deal with inconsistent scores retrieved from different CVSS scored reports across different repository sources. The reconciled scores are used as training ground for our ML models, together with vulnerability reports. Then we streamline score-computation by using a ML pipeline that classifies these instances considering various CVSS-metric labels. CVSS metrics from different CVSS versions are stored in a knowledge base whereby the corresponding metric-set is retrieved through user's query. The same goes for measurements and severity scales. In doing so, one can select any CVSS version to compute corresponding score and vector for vulnerability instances. The word "pipeline" here suggests a series of steps chained together by integrating ML cycles, where each cycle involves obtaining the data (both the training instances and the classification measurements into categorised metric classes), pre-processing it, training/testing it on a machine-learning algorithm and finally obtaining some output (in the form of a severity-score prediction).

4.1 Majority Voting for Inconsistent Scores

A small percentage of score records in NVD is assumed to have errors [13]. Therefore, relying upon NVD scores alone can bring bias in vulnerability assessment. Besides statistical vulnerability patterns mined from CVE reports, other data sources like vendors and third-party security analysers (e.g. ICS CERT and VulDB) may provide different perspectives for vulnerability scoring. In our proposed severity-computation pipeline, we use the score that the majority data-sources in the pipeline agree on as an estimate of the true score [15]. If only two data-sources provide scores, and these scores are inconsistent, then we take the average of the scores.

Here, we give an example using the vulnerability instance *CVE-2014-0754* which is assigned a CVSS V2 base-score 10.0 by NVD, VulDB, as well as ICS CERT with a vector *AV:N/AC:L/Au:N/C:C/I:C/A:C*. Yet, a different score 9.3 is assigned by the vendor *Schneider Electric* with a vector *AV:N/AC:M/Au:N/C:C/I:C/A:C*. The inconsistence occurs due to different measurements for *Access Complexity* of this instance, whereby *Schneider Electric* assigns medium complexity, while the other three parties assign low complexity. Using our majority voting approach, we choose a final score of 10.0 as true score. Another example is the vulnerability instance *CVE-2018-7791* for which a CVSS V3 base-score of 9.8 is assigned by NVD and vendor Schneider Electric with the vector *AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H*. Yet, ICS CERT assigns this vulnerability with score 7.7 with the vector *AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L*. Similarly, the inconsistence comes as a result of different measurement for attack complexity. Using our majority voting approach, we choose a final score of 9.8 as the true score.

4.2 Vulnerability Severity Computing

Our proposed machine-learning pipeline supports the classification of cybersecurity data, and fills-up missing scores in vulnerability reports using text-mining techniques [14]. We retrieve new vulnerability instances from online cybersecurity databases, together with a list of vulnerability descriptions for these instances. To classify new vulnerability instances into different CVSS-metric categories, we build a machine-learning classifier and train the classifier based on some historical data.

Formally, a dataset D with n vulnerability instance where each instance (x_i, Y_i) $(0 < i \leq N)$ has a vulnerability report x_i and a ground truth vector Y_i. Given a set of CVSS metrics $m = [m_1, \ldots, m_j, \ldots, m_M]$ $(0 < j \leq M)$, then $Y_i = [Y_i^{(m_1)}, \ldots, Y_i^{(m_j)}, \ldots, Y_i^{(m_M)}]$ where $Y_i^{(m_j)} \in \{l_1, \ldots, l_k, \ldots, l_K\}$ $(0 < k \leq K)$ for a metric m_j with K classes. Here we take CVSS Version 3 as an example, then CVSS metric-set $m = [AV, AC, PR, UI, S, C, I, A]$, and $Y_i =$

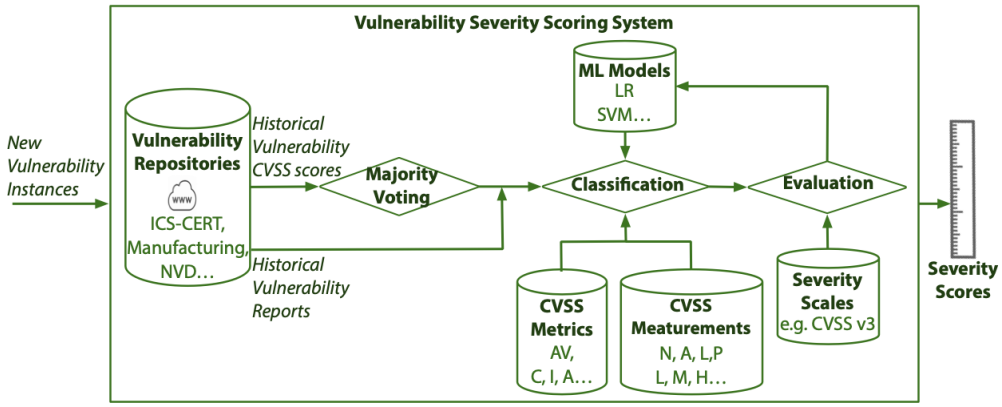

Figure 1: Vulnerability Severity Computing Pipeline

$[Y_i^{(AV)}, Y_i^{(AC)}, Y_i^{(PR)}, Y_i^{(UI)}, Y_i^{(S)}, Y_i^{(C)}, Y_i^{(I)}, Y_i^{(A)}]$ where $Y_i^{(AV)} \in \{N, A, L, P\}$, $Y_i^{(AC)} \in \{L, H\}$, $Y_i^{(PR)} \in \{N, L, H\}$, $Y_i^{(UI)} \in \{N, R\}$, $Y_i^{(S)} \in \{U, C\}$, $Y_i^{(C)} \in \{H, L, N\}$, $Y_i^{(I)} \in \{H, L, N\}$, $Y_i^{(A)} \in \{H, L, N\}$. A severity-score is computed using Algorithm 1. For a classification metric m_j with $K(K > 2)$ classes, we reduce this classification metric to several binary classification problems, while each problem discriminates between one class from the rest. So with a "one-against-all" method, we have $f(x_i) = \arg\max_k f_k(x_i)$.

Algorithm 1 Automatic Vulnerability Severity Computing

procedure SeverityComputing($\mathcal{ML}, D, m, K$)
▷ $\mathcal{ML}$ is a machine learning model $f()$, m is a set of CVSS metrics m_j $(0 < j \leq M)$ where each metric m_j has a set of K^{m_j} classes, and D is a dataset with n vulnerability instance where each instance (x_i, Y_i) $(0 < i \leq N)$ has a vulnerability report x_i and a ground truth vector Y_i.
  $N = |D|, M = |m|$
  for $j = 1, \ldots, M$ **do**
    Train(ML_j)
    $f(x)^j = \arg\max_{K^{m_j}} f_{K^{m_j}}(x)^j$
  end for
  for $i = 1, \ldots, N$ **do**
    for $j = 1, \ldots, M$ **do**
      $Z_i^{(m_j)} = f(x_I)^j$
    end for
    $Z_i = [Z_i^{(m_1)}, \ldots, Z_i^{(m_j)}, \ldots, Z_i^{(m_M)}]$
  end for
  Severity Score $S_i = CVSSCALCULATION(Z_i)$
end procedure

Using $CVSSCALCULATION(Z_i)$, CVSS measurement classification labels are further mapped to scaled attributes in order to compute severity scores. Using CVSS V3.0 as an example, *Attack-Complexity (AC)* with *high* label refers to a score attribute of 0.44, while a *low* label refers to a score attribute of 0.77.

4.3 Evaluation Metrics

For training/testing classifications, we compare predicted severity labels to their original severity label counterparts, and apply the evaluation metrics of Balanced Accuracy (adopted from [2]) and F1-score (adopted from [11]) to evaluate the machine-learning model performance, which takes into account the effect of imbalanced classes. For example, in *AccessVector(AV)* classes, sample-size in the *Network* category are much larger than sample-size in the *Physical* category. This class imbalance phenomenon is also highlighted in Table 2 later. For binary classifications such as *UserInteraction (UI)* classification, we apply the confusion matrix to compute the value of classification balanced-accuracy and F1-score. For multi-class classifications such as *AccessVector(AV)* classification, we use micro-average to calculate the average of per class evaluation, which takes into consideration the contribution of each class.

4.4 Performance Validation

We retrieved 138 331 vulnerability records with index year values ranging from 2002 to 2019 using NVD (updated till October 27th, 2020), and removed the reports that are marked as *REJECT*. We further removed the vulnerability instances that are not scored under CVSS V2 mechanisms, and set up a corpus for CVSS V2 severity computing with the remaining 127 907 vulnerability reports. We correlate these vulnerability instances with authoritative data-sources such as ICS-CERT (which provides cybersecurity experts' insights) as well as vendors' websites, to reconcile inconsistent scores. These scores are used later as ground truths for ML model training. Similarly, we removed the ones that are not scored under CVSS V3 mechanisms, and set up a corpus with 58 813 instances for CVSS V3 severity computation. Note that only a small amount of the vulnerability instances reported in and before the index year 2015 are scored under CVSS V3, which sums up to 4 958 items.

We built a machine-learning pipeline using an existing Python package *pipeline* from *Scikit-learn* library, to automate the machine-learning workflow like data processing and features extraction. Such a pipeline structure also allows the transformation of severity scores using different CVSS-version mechanisms (e.g. CVSS V2 and CVSS V3) efficiently. To do so, we first conducted some data pre-processing like tokenisation and feature extractions on

vulnerability reports from NVD. More specifically, *CountVectorizer* and *TdidfTransforer* utilities are used for vectorisation and TF-IDF (referring to Term Frequency-Inverse Document Frequency) value computation, based on which we created a TF-IDF sparse matrix using a sequence of word features. Then we adopted *train_test_split* utility to divide our data records into a 75% training dataset and a 25% validation dataset, randomly.

Table 1: Evaluation of CVSS Categorisation

CVSS-Metric	Balanced Accuracy	Micro F1-Score
V2 AccessVector(AV)	80.87%	95.76%
V2 AccessComplexity(AC)	63.68%	83.63%
V2 Authentication(Au)	56.34%	95.00%
V2 ConfidentialityImpact(C)	81.03%	82.98%
V2 IntegrityImpact(I)	82.40%	84.60%
V2 AvailabilityImpact(A)	80.12%	81.08%
V3 AttackVector(AV)	75.92%	93.68%
V3 AttackComplexity(AC)	78.78%	95.58%
V3 PrivilegesRequired(PR)	78.79%	90.71%
V3 UserInteraction(UI)	93.45%	94.13%
V3 Scope(S)	93.65%	97.48%
V3 ConfidentialityImpact(C)	88.36%	91.46%
V3 IntegrityImpact(I)	90.58%	92.02%
V3 AvailabilityImpact(A)	75.75%	93.01%

Machine-learning algorithms are then selected and trained with extracted features. Such classifiers are employed to categorise new vulnerability instances in order to generate severity patterns. In our case study, we present the results using *LogisticRegression* (LR) classifier. We also applied a 5-fold stratified cross-validation on the training dataset for CVSS categorisation to minimise a possible overfitting. The prediction performances of our CVSS classifier are listed in Table. 1, which look promising compared to similar CVSS classification works by Gawron et al., [5] and by Yamamoto et al., [16]. Model performance in Gawron et al. is evaluated using accuracy though, which can not reflect the true performance of unbalanced data classification. Interestingly, classifications using CVSS V3 metrics achieve an overall better performance than those using CVSS V2 metrics, even though the error-rate of CVSS V3 is superimposed by a larger number of metrics.

5 CYBER-PHYSICAL SYSTEMS VULNERABILITY ASSESSMENT

Vulnerabilities in CPS infrastructure are assessed to enumerate and rank their severity, in order to prevent threat-induced anomalies or intrusion attempts [7]. Here we present a case study of several prominent CPS components vulnerability analysis. We first query for CPS relevant vulnerabilities from online cybersecurity data-sources. Then, we compute the CVSS V3 base-scores and corresponding vectors for retrieved vulnerability instances. Finally, we perform an analysis to explore the statistical patterns of existing CPS vulnerabilities.

5.1 CPS Vulnerability Filter

We set up a query filter using PyCharm Edu python environment to retrieve CPS-relevant vulnerability instances from NVD, as well as vendor websites, and ICS CERT. The setup of the query logic is important to identify all the relevant vulnerability instances, and also to filter out possible false positives based on other keywords that distinguish them from the CPS concept. For example, in some cases using *MTU* as the only keyword might retrieve vulnerabilities either relevant to *Maximum Transfer Unit* (e.g. vulnerability instance *CVE-2005-0065*) or *Master Terminal Unit* (e.g. vulnerability instance *CVE-2015-0990*). Therefore, to retrieve the Master Terminal Unit relevant vulnerabilities, we define further keywords like *SCADA-server*. In our method, we extracted the entities of vulnerable components in CVE vulnerability reports using an open-source NER (referring to Named Entity Recognition) model [17], and then map retrieved entities with the Common Platform Enumeration (CPE) as well as vendor websites, to generate a list of terms for these components, as illustrated in Fig. 2 . These terms are combined with certain versions in interest as tags to query vulnerability instances from open-source vulnerability data sources.

We also retrieved vendor information for each extracted CPS vulnerability instance using NER, and correlate them against CPE database. Vendor names in CPE metadata may appear with variations. For example, the vendor *Schneider Electric SE* has variant forms like *'schneider-electic'*, *'chneider-electric'*, and *'schneider-electric'*. We conduct some manual checking on vendor metadata, and further optimise our search engine to detect hidden metadata for each vendor, to decrease possible false negatives.

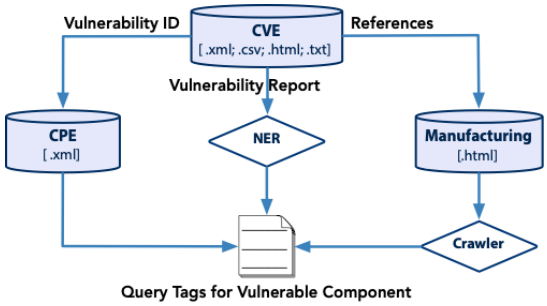

Figure 2: Generate Query Tags

5.2 CPS Vulnerabilities

Considering the nature of CPS, we define an asset component to be either software, hardware (e.g. a hard drive), or operating system (OS) (e.g. a Windows system). In CPS, hardware, software, and OS are assembled and used in different ways within CPS fabrics, which might create various binaries with potential backdoors. Outdated software might contain flaws in source code. Such flaws might be exploited by a bypass threat that is further materialised by a code-injection attack, triggered by malicious actors. CPS hardware is vulnerable by having no physical-access protection, which might be used by an attacker to gain unauthorised physical access through USB for instance. An OS that contains resource management error might be exploited by a Denial of Service (DoS) threat into a

buffer overflow attack, which might result in loss of control of this OS. Therefore, a vulnerability could be regarded as an emergent property of an asset within CPS. Combining the *Criticality* of vulnerable assets, the *Severity* of emerging *Vulnerability* instances, as well as the *Likelihood* of exploiting attacks could contribute to the computation of the asset *Vulnerability Index*. In this paper, we focus on the *Severity* score only.

We demonstrate our streamlined vulnerability-severity scoring method considering prominent CPS components such as PLCs, RTUs, MTUs and HMIs. A Programmable Logic-Controller (PLC) is a crucial CPS asset that controls industrial devices, to keep production-processes in order. A Remote Terminal Unit (RTU) transmits telemetry data from sensing devices that are associated with power physical-components to a Master Terminal Unit (MTU) system. A Human Machine Interface (HMI) provides a communication interface to visualise and monitor MTU activities and RTU information flow [7].

The results obtained from querying our database, show 89, 32, 8, and 105 reported vulnerability instances related to PLC, RTU, MTU and HMI respectively. These data are retrieved till October 27th, 2020. Although the amounts of encountered reports for CPS assets increase in the recent ten years, still the total reports are limited compared to the vulnerability instances for the commonly seen information technology systems. Some of these vulnerabilities are not scored or classified till the retrieval date, so we first use our system to fill-up any missing score or inconsistent score before further analysis.

5.3 Inconsistent Scores of CPS Vulnerabilities

Here we compare the inconsistent CVSS scores of CPS vulnerabilities, with a purpose of exploring what makes these scores inconsistent. We consider three types of scoring sources, namely NVD, ICE-CERT, and vendors.

Taking PLC relevant vulnerabilities as an example, in total, 8 CVSS V2 data records of PLC vulnerabilities are inconsistent. These vulnerability instances have IDs *CVE-2015-3938*, *CVE-2015-7937*, *CVE-2012-3037*, *CVE-2013-0664*, *CVE-2015-6462*, *CVE-2011-5007*, *CVE-2015-6461*, *CVE-2014-0754*. We also compare the CVSS v3 scores directly retrieved from NVD with scores mined from other authority-bound sources, and amend inconsistent results when taking vendors' and third-party reports into consideration. In total, 6 CVSS v3 scores of PLC vulnerability records reported in NVD are modified. These vulnerability instances have IDs *CVE-2016-8354*, *CVE-2018-10594*, *CVE-2018-7791*, *CVE-2017-14465*, *CVE-2017-14470*, *CVE-2017-6023*.

Taking MTU relevant vulnerabilities as another example, 3 CVSS V2 data records (i.e. *CVE-2014-0752*, *CVE-2014-2375*, *CVE-2015-0990*) of MTU vulnerabilities are inconsistent, and 1 CVSS V3 data record (i.e. *CVE-2020-6970*) of MTU vulnerabilities is inconsistent.

We conduct further statistical analysis on the rationale of inconsistent scores. For CVSS V2 scoring, the most potential inconsistency comes from Access Complexity metric, followed by Confidentiality Impact and Integrity Impact metrics. For CVSS V3 scoring, the most possible inconsistence also comes from Attack Complexity metric, followed by Scope and Confidentiality Impact metrics. Different cybersecurity analysis parties seem to have better agreement on Authentication metric from CVSS V2, as well as Privileges Required and User Interaction metrics from CVSS V3.

5.4 CPS Vendors

Subsequently, we generate a word-net to illustrate the main characteristics of retrieved vulnerability instances from different vendors. We select some distinct vendors for CPS vulnerabilities, namely *Schneider Electric SE* and *Siemens AG*, and extract vulnerabilities relevant to these two vendors. In total, we found 24 vulnerability instances in affected products from *Schneider Electric SE*, and 39 vulnerability instances in affected products from *Siemens AG*. Then, we conduct some text preprocessing on compiled vulnerability reports. We use *CountVectorizer* and *TdidfTransforer* from *Scikit-learn* library to extract keywords with different levels of importance. We also customised a stop-word list to remove vendor or product-version related information. Then, we use WordNetLemmatizer from the Natural Language Toolkit (NLTK) to discover top keywords in vulnerability reports in the form of word-nets. The top-50 keywords of vulnerability reports relevant to *Schneider Electric SE* and *Siemens AG* are illustrated in Fig. 3 (a) and (b), respectively. A keyword with bigger size in a word-net represents higher importance to that vendor. These two word-nets have some keywords in common, like *server*, *web*, *remote*, *denial*, etc. These common keywords highlight a trend of network-based exploits for retrieved vulnerabilities. Denial of service seems to be a common attack exploiting vulnerabilities in *Schneider Electric SE* and *Siemens AG*.

Figure 3: Top Keywords Relevant to Schneider and Siemens

Table 2: Measurements Distribution of Base Metrics

Metric	Measurement	PLC	RTU	MTU	HMI	CPS	CVE
AttackPath	Network	93.26%	100%	87.50%	84.76%	90.17%	74.35%
	AdjacentNetwork	0%	0%	0%	0.95%	0.43%	22.57%
	Local	5.62%	0%	12.50%	13.33%	8.55%	2.01%
	Physical	1.12%	0%	0%	0.95%	0.85%	1.06%
AttackComplexity	Low	97.75%	100%	100%	99.05%	98.72%	91.21%
	High	2.25%	0%	0%	0.95%	1.28%	8.79%
PrivilegesRequired	None	95.51%	78.13%	100%	92.38%	91.45%	69.55%
	Low	4.49%	12.50%	0%	7.62%	7.26%	25.18%
	High	0%	9.37%	0%	0%	1.28%	5.28%
UserInteraction	None	83.14%	100%	87.50%	67.62%	78.63%	62.80%
	Required	16.85%	0%	12.50%	32.38%	21.37%	37.20%
ScopeChange	Unchanged	92.13%	100%	100%	85.71%	90.60%	83.64%
	Changed	7.87%	0%	0%	14.29%	9.40%	16.36%
ConfidentialityImpact	None	43.82%	21.87%	50.00%	17.14%	26.92%	22.15%
	Low	7.87%	0%	0%	10.48%	8.97%	19.10%
	High	48.31%	78.13%	15.00%	72.38%	64.10%	58.75%
IntegrityImpact	None	42.70%	46.88%	50.00%	39.05%	43.16%	31.14%
	Low	7.87%	0%	0%	10.48%	7.69%	17.20%
	High	49.45%	53.12%	50.00%	50.48%	49.15%	51.66%
AvailabilityImpact	None	13.48%	28.13%	37.50%	29.52%	22.65%	38.22%
	Low	0%	0%	0%	1.90%	0.85%	2.30%
	High	86.52%	71.87%	62.50%	68.57%	74.36%	61.19%

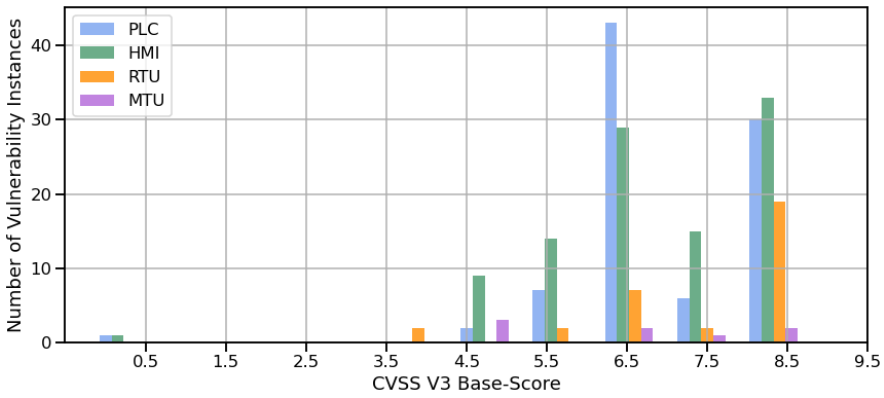

Figure 4: CVSS Version3 Base-Scores Distribution of CPS Vulnerabilities (2002-2020)

5.5 Characteristic Analysis for CPS Vulnerabilities

For retrieved CPS vulnerabilities, we re-compute their scores using our scoring system introduced in Section 4. Then, we examine the diversity of their sub-scores to better understand the metrics composition introduced in CVSS V3 scores, through the analysis of sub-vectors. Here, we mainly emphasise the base dimensions. Base attributes of CPS vulnerabilities can be partially investigated by the rate of actually assigned values to Exploitability, Scope and Impact metric measurements, as presented in Table 2. We list the attributes for each CPS asset separately in Columns 3-6, list the attributes for CPS on average in Column 7, and list the overall percentage of published CVE reports in Column 8.

Compared to the average characteristics of their instances in CVE, the exploitability sub-vectors of CPS vulnerability measurements indicate that most (90.17%) of the attacks are Network-based, especially for RTU vulnerability instances. Adjacent network-based attacks have limited presences in CPS. In contrast, local attacks have more presences in CPS. Most CPS vulnerability instances, i.e. 98.72%, can be exploited by low-skilled attackers, given the observed attack-complexity patterns, and the assigned "low" AttackComplexity measurement. Limited CPS vulnerability instances need high privileges. 21.37% of exploits require user interactions. Only 9.40% of exploits in CPS result in changed scope and possibly more severe impact. Compared to exploitability and scope sub-vectors, the

impact sub-vectors of CPS vulnerability instances present higher diversity among possible impact values. It is also noticeable that CPS vulnerability instances bring more impact to the confidentiality or availability than the integrity of the vulnerable components.

We compute the severity base-scores of retrieved CPS vulnerabilities using CVSS V3, as illustrated in Fig. 4. Both HMI and PLC vulnerabilities base-scores have high presences at 6.5 and 8.5. Base-scores of RTU vulnerabilities have a top presence at 8.5. MTU vulnerabilities have more variant distributions in the score threshold [4.5-8.5]. These vulnerability instances show an average CVSS V3 Base-Scores of 7.54, 8.00, 6.88 and 7.51 for PLC, RTU, MTU and HMI respectively. We further map these CVSS V3 base-scores to the CVSS qualitative severity rating scale[2]. Most of the CPS vulnerability instances have either *medium* (*[4.0-6.9]*) or *high* (*[7.0-8.9]*) severity.

6 CONCLUSION AND FUTURE WORKS

Understanding, pinpointing and scaling vulnerabilities in CPS networks are challenging tasks, yet vital for cybersecurity purposes. In this paper, we introduced a streamlined vulnerability-severity scoring system based on the industrial standard CVSS mechanisms, to discover the severity of a vulnerability instance. Our system computes severity scores for different CVSS versions, which improves the consistency of vulnerability analysis and thus related prioritisation of remediation-tasks. We applied majority voting to reconcile inconsistent scores for the same vulnerabilities provided by different cybersecurity-repository sources, and used these reconciled vulnerability instances to train our machine-learning based scoring model. Our model achieves good performance using balanced accuracy and micro F1-score evaluation metrics. Finally, we provided a detailed evaluation of our proposed scoring approach through a case study of CPS vulnerability instances, using multiple online cybersecurity data sources. In this case study, CPS vulnerability instances are retrieved using a query filter with customised keywords and logic. Using these extracted vulnerability instances, we analyse the characteristics of CPS vulnerabilities, in comparison with the general characteristics of existing vulnerabilities in CVE repository. Typically, reported CPS vulnerability instances are mostly network-based or local-based. Exploiting a CPS vulnerability generally expects low attack complexity, low privilege, and low user interaction. A successful attack on CPS vulnerabilities probably brings a high impact on the availability of the related CPS asset. The scores of reported CPS vulnerabilities present a centralised distribution in CVSS V3 base-score scale, especially in the score scales of medium- and high-severity.

As part of our future works, we can further optimise our scoring system by adjusting the tie of majority voting under experts' supervision. Such reliability evaluation can start with inviting security experts to set an initial value, and then apply some computational intelligence techniques to adjust it further. Another possible extension may use the weighted arithmetic mean of different scores from several sources, while measuring the weights by evaluating the reliability of the score sources. We also plan to explore the differences between applying majority voting to the sub-metrics of CVSS, with the method used in this paper whereby majority voting is applied to the CVSS base-scores directly.

[2]https://www.first.org/cvss/v3.1/specification-document

REFERENCES

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

[2] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. 2010. The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*. IEEE, 3121–3124.

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

[4] Steve Christey and Brian Martin. 2013. Buying into the bias: Why vulnerability statistics suck. *BlackHat, Las Vegas, USA, Tech. Rep* 1 (2013).

[5] Marian Gawron, Feng Cheng, and Christoph Meinel. 2017. Automatic vulnerability classification using machine learning. In *International Conference on Risks and Security of Internet and Systems*. Springer, 3–17.

[6] Dan Geer and Michael Roytman. 2013. Measuring vs. modeling. *; login:: the magazine of USENIX & SAGE* 38, 6 (2013), 64–67.

[7] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. 2017. Cyber-physical systems security—A survey. *IEEE Internet of Things Journal* 4, 6 (2017), 1802–1831.

[8] Pontus Johnson, Robert Lagerström, Mathias Ekstedt, and Ulrik Franke. 2016. Can the common vulnerability scoring system be trusted? a bayesian analysis. *IEEE Transactions on Dependable and Secure Computing* 15, 6 (2016), 1002–1015.

[9] Atefeh Khazaei, Mohammad Ghasemzadeh, and Vali Derhami. 2016. An automatic method for CVSS score prediction using vulnerabilities description. *Journal of Intelligent & Fuzzy Systems* 30, 1 (2016), 89–96.

[10] Marcus Pendleton, Richard Garcia-Lebron, Jin-Hee Cho, and Shouhuai Xu. 2016. A survey on systems security metrics. *ACM Computing Surveys (CSUR)* 49, 4 (2016), 1–35.

[11] David Martin Powers. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. (2011).

[12] Jukka Ruohonen. 2019. A look at the time delays in CVSS vulnerability scoring. *Applied Computing and Informatics* 15, 2 (2019), 129–135.

[13] Karen Scarfone and Peter Mell. 2009. An analysis of CVSS version 2 vulnerability scoring. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, 516–525.

[14] Georgios Spanos, Lefteris Angelis, and Dimitrios Toloudis. 2017. Assessment of vulnerability severity using text mining. In *Proceedings of the 21st Pan-Hellenic Conference on Informatics*. 1–6.

[15] Dapeng Tao, Jun Cheng, Zhengtao Yu, Kun Yue, and Lizhen Wang. 2018. Domain-weighted majority voting for crowdsourcing. *IEEE transactions on neural networks and learning systems* 30, 1 (2018), 163–174.

[16] Yasuhiro Yamamoto, Daisuke Miyamoto, and Masaya Nakayama. 2015. Text-mining approach for estimating vulnerability score. In *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. IEEE, 67–73.

[17] Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345* (2017).

[18] Ziyun Zhu and Tudor Dumitraş. 2016. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 767–778.