

## VULNERABILITY ANALYSIS FOR CRITICAL INFRASTRUCTURES



DOCTORAL DISSERTATION

VULNERABILITY ANALYSIS FOR CRITICAL  
INFRASTRUCTURES

YUNING JIANG  
*Informatics*



UNIVERSITY  
OF SKÖVDE

Yuning Jiang, 2022

Doctoral Dissertation

*Title:* Vulnerability Analysis for Critical Infrastructures

University of Skövde, Sweden  
[www.his.se](http://www.his.se)

*Printer:* Stema Specialtryck AB, Borås

ISBN 978-91-987906-0-3  
Dissertation Series No. 46 (2022)

*“Nothing has such power to broaden the mind as the ability to investigate systematically and truly all that comes under thy observation in life.”*

– Marcus Aurelius



# ABSTRACT

The rapid advances in information and communication technology enable a shift from diverse systems empowered mainly by either hardware or software to cyber-physical systems (CPSs) that drive critical infrastructures (CIs), such as energy and manufacturing systems. However, alongside the expected enhancements in efficiency and reliability, the induced connectivity exposes these CIs to cyberattacks such as the *Stuxnet* and *WannaCry* ransomware cyber incidents. Therefore, the need to improve cybersecurity expectations of CIs through vulnerability assessments cannot be overstated. Yet, CI cybersecurity has intrinsic challenges due to the convergence of information technology (IT) and operational technology (OT) as well as the cross-layer dependencies inherent to CPS based CIs. Different IT and OT security terminologies also lead to ambiguities induced by knowledge gaps in CI cybersecurity. Moreover, current vulnerability-assessment processes in CIs are mostly subjective and human-centered. The imprecise nature of manual vulnerability assessment operations and the massive volume of data cause an unbearable burden for security analysts. Latest advances in cybersecurity solutions based on machine-learning promise to shift such burden to digital alternatives. Nevertheless, the heterogeneity, diversity and information gaps in existing vulnerability data repositories hamper accurate assessments anticipated by these ML-based approaches. To address these issues, this thesis presents a comprehensive approach that unleashes the power of ML advances while still involving human operators in assessing cybersecurity vulnerabilities within deployed CI networks.

Specifically, this thesis proposes data-driven cybersecurity indicators to bridge vulnerability management gaps induced by ad-hoc and subjective auditing processes as well as to increase the level of automation in vulnerability analysis. The proposed methodology follows design science research principles to support the development and validation of scientifically-sound artifacts. More specifically, the proposed data-driven cybersecurity architecture orchestrates a range of modules that include: (i) a vulnerability data model that captures a variety of publicly accessible cybersecurity-related data sources; (ii) an ensemble-based ML pipeline method that self-adjusts to the best learning models for given cybersecurity tasks; and (iii) a knowledge taxonomy and its instantiated power grid and manufacturing models that capture CI common semantics of cyber-physical functional dependencies across CI networks in critical societal domains.

This research contributes data-driven vulnerability analysis approaches that bridge the knowledge gaps among different security functions, such as vulnerability management through related reports analysis. This thesis also correlates vulnerability analysis findings to coordinate mitigation responses in complex CIs. More specifically, the vulnerability data model expands the vulnerability knowledge scope and curates meaningful contexts for vulnerability analysis processes. The proposed ML methods fill information gaps in vulnerability repositories using curated data while further streamlining vulnerability assessment processes. Moreover, the CI security taxonomy provides disciplined and coherent support to specify and group semantically-related components and coordination mechanisms to harness the notorious complexity of CI networks such as those

prevalent in power grids and manufacturing infrastructures. These approaches learn through interactive processes to proactively detect and analyze vulnerabilities while facilitating actionable insights for security actors to make informed decisions.

keywords: critical infrastructure cybersecurity, vulnerability assessment, vulnerability quantification, computational intelligence in cybersecurity, cyber-physical system

# SAMMANFATTNING

De snabba framstegen inom informations- och kommunikationsteknologi möjliggör ett skifte från olika separata system, huvudsakligen drivna av antingen hård- eller mjukvara, till stora cyberfysiska system (CPS) som driver kritiska infrastrukturer såsom energi- och tillverkningssystem. Men vid sidan av förväntade förbättringar, i form av ökad effektivitet och tillförlitlighet, så ökas även exponeringen för cyberattacker som en följd av den ökade uppkopplingen. Cyberincidenter som t ex Stuxnet och WannaCry ransomware illustrerar detta. Behovet av att förbättra cybersäkerheten hos kritiska infrastrukturer, genom analys och bedömning av deras sårbarheter, kan därför inte överskattas. Cybersäkerhet hos kritiska infrastrukturer har dock inneboende utmaningar, dels på grund av att informationsteknologi (IT) och operativ teknologi (OT) konvergerar, och dels på grund av beroenden mellan olika lager hos systemet. Olika terminologi för säkerhet inom IT och OT bidrar också till otydlighet och kunskapsgap när det gäller cybersäkerhet hos kritiska infrastrukturer. Nuvarande processer för att utföra sårbarhetsbedömningar i den här typen av system är dessutom mestadels subjektiva och mäniskocentrerade. Den oprecisa karaktären hos manuella sårbarhetsbedömningar och den enorma mängden data att analysera lägger en omöjlig börd på säkerhetsanalytiker. De senaste framstegen inom maskininlärning (ML) för ML-baserade cybersäkerhetslösningar är lovande och kan komma att flytta den stora bördan till digitala lösningar. Utmaningar i form av heterogenitet, diversitet och otillräcklig information i de datalager med tillgänglig information om sårbarheter sätter dock upp hinder för att göra de korrekta bedömningarna som förväntas av ML-baserade tillvägagångssätt. Därför presenteras i denna avhandling ett heltäckande tillvägagångssätt för att utnyttja de framsteg som gjorts inom ML samtidigt som mänskliga operatörer involveras i bedömningen av cybersäkerhetssårbarheter inom nätförkortning för distribuerade kritiska infrastrukturer.

Avhandlingen presenterar datadrivna indikatorer för cybersäkerhet. Syftet är dels att överbrygga de luckor i sårbarhetshanteringens som kommer av ad-hoc och subjektiva processer, och dels att öka automatiseringen av sårbarhetsanalys. Metodiken i arbetet följer design science principerna för utveckling och validering av vetenskapligt sunda artefakter. Mer specifikt innehåller den datadrivna cybersäkerhetsarkitektur som föreslås här en rad moduler som inkluderar: (i) en datamodell över sårbarheter som täcker in en mängd olika och allmänt tillgängliga cybersäkerhetsrelaterade datakällor; (ii) en ensemblebaserad ML-pipeline som anpassar sig till de bästa inlärningsmodellerna för givna cybersäkerhetsuppgifter; och (iii) en taxonomi, med instansierade modeller för kraftnät och för tillverkning, vilken fångar kritiska infrastrukturers gemensamma semantik för cyberfysiska funktionella beroenden inom nätförkortning hos kritiska samhällsfunktioner.

Forskningen bidrar med datadrivna metoder för sårbarhetsanalys. Metoderna överbrygger kunskapslyftor mellan olika funktioner för säkerhet, såsom sårbarhetshantering, genom analys av relaterade rapporter. Avhandlingen korrelerar också resultat inom sårbarhetsanalys för att koordinera åtgärder som syftar till att begränsa sårbarheten hos komplex, kritisk infrastruktur. Mer specifikt utökar sårbarhetsdatamodellen kunskapen

om sårbarheterna och skapar meningsfulla sammanhang för processer för sårbarhetsanalys. De föreslagna ML-metoderna fyller igen luckor i information om sårbarheter med hjälp av kurerað data samtidigt som processerna för sårbarhetsbedömning effektiviseras ytterligare. Dessutom ger taxonomin för säkerhet inom kritiska infrastrukturer ett klart och sammanhängande stöd för att specificera och gruppera semantiskt relaterade komponenter och mekanismer i syfte att hantera komplexiteten hos kritiska infrastrukturers nätverk såsom de som är vanliga i kraftnät och tillverkningsinfrastruktur. Metoderna lär sig, genom interaktiva processer, att proaktivt upptäcka och analysera sårbarheter samtidigt som de underlättar för säkerhetsaktörer att agera på informationen och fatta välgrundade beslut.

# ACKNOWLEDGEMENTS

I came to Sweden after having studied in the UK and France. Although I grew up in several quite different provinces in China, moving to a new country was terrifying. Yet after five years, Skövde has become a place where my intellectual identities found enlightenment and meaning. Many times I stood on *Billingen* and overlooked the city, watching how the lights were beginning to pierce brightly into the soft and darkening sky, one star after the other. These stars lighted my way home and left me with a true sense of belonging. For this, I am genuinely grateful, and want to take the opportunity to say thank you to many wonderful people I have met during this Ph.D. journey.

First and foremost, I would like to express my deepest thanks to my main supervisor *Prof. Yacine Atif*, and four co-supervisors *Associate Prof. Jianguo Ding*, *Prof. Manfred A. Jeusfeld*, *Associate Prof. Birgitta Lindström* and *PhD Christoffer Brax*. Their expertise and immense knowledge in different fields supported this interdisciplinary study. Moreover, their kindness, patience, encouragement, and wisdom are much more than the guidance in my Ph.D studies, as they significantly impacted my personal growth. *Yacine*, I would like to express my deepest gratitude to you for always encouraging my research and providing insightful comments. Your elegant definition of “*re-search*” and your wise advice helped me grow as a researcher. I am also grateful for your incredible support in writing revisions and presentation rehearsals. *Ding*, I am deeply indebted to your generous feedback and some extra push forward when needed. I enjoyed our discussions on the correlations between complex systems and ancient Chinese philosophy in the *Book of Changes*. I am also truly thankful for your kind advice on my personal development. *Manfred*, I am extremely grateful for your constructive feedback and genuine support throughout my Ph.D. studies. Your positive attitude inspired me to look on the brighter side. So many times in our discussions, you improved my ideas on the fly and helped me form those ideas into practical solutions. *Birgitta*, I would like to express my deep appreciation for your continuous support and willingness to be my go-to option whenever I have formality issues. It is a great honor to work with you. I will never forget my first Swedish Christmas Eve in your place, the memory of which I will always cherish. *Christoffer*, I am particularly grateful for your inspiration along the journey and for introducing me to the practical side of cybersecurity. I always learn something new when I speak to you. I am also thankful for your generous feedback on my career development.

My sincere gratitude extends to *Prof. Rose-Mharie Åhlfeldt*, *Associate Prof. Shahid Raza*, and *Associate Prof. Tomas Olovsson*, who all have been great opponents at my seminars and provided valuable feedback.

I would also like to acknowledge and thank the organizations and all the respondents involved in different stages of this research. I am incredibly grateful to all the members of the *ELVIRA* (threat modeling and resilience management for critical infrastructures) project, for all your generous input in the group meetings. I am also thankful for the opportunity to transform parts of the *ELVIRA* project results into a master’s level course, together with the group members. In addition, I would like to express my gratitude to *Christine Mulder* in *AFSI* for generous help with research applications and encourage-

ment to make my research findings more accessible to the general public.

Furthermore, I would like to send my thanks to all the people I have had great pleasure to teach with. Thank you for showing me great enthusiasm for education and shaping my teaching perspective. I especially enjoyed all the fun lab supervisions together with *Sara Mahmoud* and *András Márki*. *Sara*, my neighbor and friend, thank you for always finding time for me when I disturbed you for random idea discussions, and for enormous warm hugs and positive energy whenever needed. *András*, my workout buddy and friend, thank you for the extra push when needed, and the laugh when we failed the yoga wheel pose.

I would like to send my gratitude to the University of Skövde for giving me the opportunity to come as a Ph.D. student. This life-altering experience would be much less fun without the freedom I have been given and the support I have always received. My gratitude extends to all the present and past colleagues in the DRTS group. Thank you, *Sten, Jörgen, Marcus, Jonas* and *Loreto* - I learned a lot in our group discussions. I must also thank the colleagues in the Information System group for inviting me to join their Friday morning *fikas*.

There are several fellow Ph.D students (some of whom have already graduated) that I would like to thank for the treasured memories during *fikas*, dry runs, and many other gatherings: *Fernando, Ari, Kristens, Hanife, Elio, Niclas, Navoda, Nafise, Alice, Niko, Vipul, Erik L, Joakim, Sutor* and *Kelly*. In addition, I want to send my gratitude to *Juhee* who I shared many of those precious moments with. Thank you all for the friendship along the years, for unwavering your personal and professional support.

Next, I want to thank my friends who cannot be with me physically, but always available for listening to and, at times, having to tolerate me over the phone in the past years. Special thanks go to *ChenXing, Ge* (and the little one, *Erchen*), *Ariel, YongJuan, JiaYang* and *Jie*. Without you I cannot imagine how to cope with the Ph.D. stress alone. Then, my heartfelt gratitude goes to *Stefan Knutsson* for enormous support, company, and believing in me, and for memories during Christmas and midsummer on the beautiful west coast with the lovely family.

Finally, none of this could have happened without my family, thank you for being my rock through the ups and downs of this Ph.D. journey. My father *WenXiang Jiang* and mother *QunWan OuYang*, thank you for providing me with opportunities to explore the world, for sharing tons of books and delightful traveling adventures, for reminding me to look at the bigger picture and enjoy life when I was troubled by little things. I could not imagine better parents. Then my grandparents, aunts, uncles and cousins, thank you for your love, trust and continuous encouragement. Special thanks to my cousin *YunPeng Li* who always stands by and shares wonderful traveling memories with me.

# PUBLICATIONS

This thesis is a monograph, while supported from the listed publications:

## HIGH RELEVANCE

- I. Yuning Jiang, Manfred Jeusfeld, Yacine Atif, Jianguo Ding, Christoffer Brax, and Eva Nero (2018b). “A Language and Repository for Cyber Security of Smart Grids.” In: *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, pp. 164–170.
- II. Yuning Jiang, Yacine Atif, and Jianguo Ding (2019). “Cyber-Physical Systems Security Based on a Cross-Linked and Correlated Vulnerability Database.” In: *International Conference on Critical Information Infrastructures Security*. Springer, pp. 71–82.
- III. 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*, pp. 1–8.
- IV. Yuning Jiang, Manfred Jeusfeld, and Jianguo Ding (2021). “Evaluating the Data Inconsistency of Open-Source Vulnerability Repositories.” In: *The 16th International Conference on Availability, Reliability and Security*, pp. 1–10.
- V. Yuning Jiang and Yacine Atif (2021). “A Selective Ensemble Model for Cognitive Cybersecurity Analysis.” In: *Journal of Network and Computer Applications* 193, p. 103210.
- VI. Yuning Jiang, Manfred Jeusfeld, and Jianguo Ding (2022). “Model-Based Cybersecurity Analysis: Extending Enterprise Modeling to Critical Infrastructure Cybersecurity.” In: *Journal of Business & Information Systems Engineering (Revision Submitted)*.
- VII. Yuning Jiang and Yacine Atif (2022). “Towards Automatic Discovery and Assessment of Vulnerability Severity in Cyber-Physical Systems.” In: *Journal of ARRAY (Available at SSRN 4019226)*.

## LOW RELEVANCE

- VIII. Yuning Jiang, Yacine Atif, Jianguo Ding, and Wei Wang (2019). “A Semantic Framework with Humans in the Loop for Vulnerability-Assessment in Cyber-Physical Production Systems.” In: *International Conference on Risks and Security of Internet and Systems*. Springer, pp. 128–143.



# CONTENTS

<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Critical Infrastructure Cybersecurity .....	1
1.2 Problem Description.....	4
1.3 Research Aim and Objectives.....	7
1.4 Research Questions.....	8
1.5 Thesis Statement.....	10
1.6 Contributions.....	13
1.6.1 Identifying Problems and Challenges.....	13
1.6.2 Expanding Vulnerability Knowledge Scope .....	14
1.6.3 Bridging the Information Gap .....	14
1.6.4 Benchmarking CI Vulnerability .....	14
1.6.5 Streamlining Vulnerability Analysis .....	15
1.6.6 Author Contributions .....	15
1.7 Research Scope.....	15
1.7.1 Key Research Elements.....	15
1.7.2 Research Area .....	17
1.7.3 Research Focus and Delimitation .....	18
1.8 Thesis Structure.....	20
<b>2 KEY CONCEPTS AND BACKGROUND KNOWLEDGE</b>	<b>25</b>
2.1 CI Cybersecurity Concepts .....	25
2.1.1 Component, Asset and System.....	26
2.1.2 Critical Infrastructure .....	28
2.1.3 Cyber-Physical System Characteristic .....	29
2.1.4 IT Security and OT Security.....	30
2.1.5 Power Grid Security and Manufacturing Security.....	32
2.2 Vulnerability Data Source Taxonomies .....	34
2.2.1 Scoring Mechanism .....	34
2.2.2 Enumerations.....	37
2.2.3 Vulnerability Repository .....	40

2.3	Vulnerability Lifecycle.....	44
<b>3</b>	<b>DATA DRIVEN VULNERABILITY ANALYSIS</b>	<b>49</b>
3.1	Vulnerability Data Preparation.....	49
3.1.1	Correlations among Vulnerability Data.....	49
3.1.2	Vulnerability Data Quality .....	51
3.2	Machine Learning for Vulnerability Analytic .....	52
3.2.1	Machine Learning Techniques for Vulnerability Assessment.....	53
3.2.2	Ensemble Method .....	56
3.3	System-Wide Vulnerability Analysis .....	57
3.3.1	Semantic Model .....	58
3.3.2	Reference Architecture .....	59
3.3.3	Cyber-Physical Dependence Analysis .....	60
<b>4</b>	<b>RESEARCH METHODS</b>	<b>65</b>
4.1	Research Paradigm and Methodology.....	65
4.1.1	Design Science Research Paradigm .....	66
4.1.2	Design Science Methodology .....	67
4.1.3	Research Methods.....	68
4.1.4	Data Generation Methods .....	70
4.2	Design Science Research Process .....	70
4.2.1	Research Process 1: Exploratory Studies .....	72
4.2.2	Research Process 2: Vulnerability Data Curation .....	73
4.2.3	Research Process 3: Bridging the Data Gap.....	74
4.2.4	Research Process 4: Vulnerability Analysis for Critical Infrastructures	76
4.3	Discussions on Trustworthiness of the Research .....	78
4.3.1	Repeatability And Reproducibility .....	78
4.3.2	Validity and Generalization .....	79
4.3.3	Usefulness and Usability .....	79
4.3.4	Ethical Concern.....	80
4.3.5	Ontology Evaluation Metrics.....	81
<b>5</b>	<b>CHALLENGE IDENTIFICATION AND BASELINE STUDIES</b>	<b>85</b>
5.1	Before the Baseline Study.....	85
5.1.1	Data Identification .....	85
5.1.2	Data Collection .....	88
5.2	Study I: Diversity and Heterogeneity of Vulnerability Data .....	89

5.3	Study II: Industry Survey Analysis on User Expectation of Vulnerability Data Sources .....	94
5.4	Study III: Vulnerability Data Quality Assessment in Actual CI System .....	96
5.4.1	Metric Definition and Measurements.....	96
5.4.2	Data Inconsistency Analysis Process.....	98
5.4.3	Investigated CI System .....	99
5.4.4	Data Sources .....	100
5.4.5	Analysis of Retrieved Vulnerabilities .....	101
5.4.6	Data Inconsistencies in Affected Products .....	101
5.4.7	Data Inconsistencies in Weakness Type and CVSS Impact .....	103
5.4.8	Evaluation through interviews.....	106
5.5	Conclusion .....	108
<b>6</b>	<b>VULNERABILITY DATA CORRELATION AND CONSOLIDATION</b>	<b>113</b>
6.1	Artifact I: Vulnerability Data Model and Query Generation Method .....	114
6.1.1	Overall Structure and Common Data Model .....	114
6.1.2	Data Correlation Using Common Tags .....	116
6.1.3	Data Consolidation.....	119
6.1.4	Query Tag Generator.....	121
6.1.5	Prototype Database Deployment .....	123
6.2	Study IV: CI Vulnerability Trend Analysis Using Correlated Database .....	124
6.2.1	Study Setup .....	124
6.2.2	Study Results .....	125
6.3	Study V: Performance Trade-off between the Instantiated Vulnerability Database and the State-of-the-Art.....	129
6.4	Conclusion .....	131
<b>7</b>	<b>VULNERABILITY DATA GAP ANALYSIS</b>	<b>135</b>
7.1	Introduction .....	136
7.1.1	Bridging Vulnerability Information Gaps .....	136
7.1.2	Enhancing Vulnerability Analysis .....	137
7.2	Text Mining.....	139
7.2.1	Text Mining Process .....	139
7.2.2	Machine Learning Algorithms .....	142
7.2.3	Validation Metrics for Classification Algorithms .....	143
7.3	Artifact II-A: Discovering Vulnerability Severity Under CVSS Metrics .....	146
7.3.1	System Overview .....	146
7.3.2	Majority Voting for Inconsistent Scores .....	147

7.3.3	Vulnerability Severity Computing .....	147
7.3.4	Study VI: Experiment on CVSS Categorization Evaluation .....	149
7.3.5	Study VII: Computing Severity Scores for CI Vulnerabilities .....	151
7.4	Artifact II-B: Categorizing Weakness.....	152
7.4.1	Handling CWE Tree Hierarchical Structure .....	152
7.4.2	Study VIII: Experiment on Weakness Categorization Evaluation .....	154
7.5	Artifact II-C: A Selective Ensemble for Vulnerability Assessment .....	157
7.5.1	Overview .....	157
7.5.2	Candidate Metrics .....	158
7.5.3	Model Selection and Ensemble .....	159
7.6	Study IX: Experiment on the Selective Ensemble Method Evaluation .....	161
7.6.1	Data Sets .....	161
7.6.2	Experiment Setup .....	162
7.6.3	Threat Categorization Results .....	163
7.6.4	CVSS Severity Categorization Results .....	165
7.7	Conclusion .....	169
<b>8</b>	<b>MODELING CRITICAL INFRASTRUCTURE FOR VULNERABILITY ANALYSIS</b>	<b>173</b>
8.1	Artifact III-A: A Taxonomy for Critical Infrastructure Vulnerability Analysis	173
8.1.1	Modeling Cyber-Physical System Based Critical Infrastructure .....	175
8.1.2	Modeling Vulnerability and Other Security Related Objects.....	178
8.1.3	Modeling Stakeholder Actors .....	179
8.2	Artifact III-B: Cyber and Cyber-Physical Functional Dependence.....	180
8.2.1	Dependence Rule.....	180
8.2.2	Cascading Modeling and Criticality Analysis .....	182
8.3	Artifact III-C: Reference Models of Power Grids.....	183
8.3.1	Public Internet and Other Networks .....	184
8.3.2	Office, Engineering, and Security Operating Center Network .....	185
8.3.3	Control Center Network .....	186
8.3.4	Substation Network.....	186
8.3.5	Power-Grid Substation .....	187
8.3.6	Data Asset Identification .....	188
8.4	Study X: Application and Evaluation in Power Grid.....	192
8.4.1	Instantiated Power-Grid Models .....	193
8.4.2	Model-Based Dependence Analysis and Cascade Modeling .....	194
8.4.3	Extending the Taxonomy to Model Real Power Grid .....	196

8.5 Conclusion .....	197
<b>9 STREAMLINING VULNERABILITY ANALYSIS FOR CRITICAL INFRASTRUCTURES</b>	<b>201</b>
9.1 Artifact IV: Vulnerability Analysis Orchestration Method for Critical Infrastructures .....	201
9.2 Study XI: Application and Evaluation in Manufacturing .....	204
9.2.1 Prototype of Manufacturing System .....	204
9.2.2 Human Robot Collaborative Assembly System .....	206
9.2.3 Model-Based Vulnerability Assessment for Human Robot Collaborative Assembly System .....	208
9.3 Conclusion .....	209
<b>10 CONCLUSION</b>	<b>213</b>
10.1 Revisiting the Objectives and the Research Questions .....	213
10.2 Future Works .....	216
<b>APPENDICES</b>	<b>221</b>
Appendix I: Survey about User's Perspective on Vulnerability Repositories .....	221
Appendix II: Interview Questionnaire for Study III .....	226
Appendix III: Vulnerability Database Output Example in JSON.....	228
Appendix IV: Instantiated Models for <i>Study X</i> .....	234
<b>REFERENCES</b>	<b>241</b>
<b>PUBLICATIONS IN THE DISSERTATION SERIES</b>	<b>275</b>



# LIST OF FIGURES

1.1	Current subjective vulnerability analysis for critical infrastructure .....	4
1.2	Connections between research aims, questions and resulted papers .....	9
1.3	Proposed data-driven vulnerability analysis for critical infrastructure .....	11
1.4	Connections between research questions .....	12
1.5	Work flows in the proposed framework.....	13
1.6	Key research areas and elements .....	16
1.7	Related research disciplines .....	18
1.8	Scope of CI reference model and instantiation .....	19
2.1	Critical infrastructure cybersecurity concepts.....	26
2.2	System, asset and component .....	27
2.3	Layered architecture of critical infrastructures .....	28
2.4	Cyber-physical system characteristics summarized from <i>Cyber-Physical Systems - A Concept Map</i> (2022).....	30
2.5	Layered structure of manufacturing system .....	33
2.6	Threat type (assigned by <i>cvedetails.com</i> ) distributions for vulnerabilities published from 1999 to November 3, 2021.....	39
2.7	<i>CVE-2021-37172</i> as an example of published vulnerabilities in <i>CVE</i> .....	41
2.8	<i>EDB-ID 44655</i> as an example of published exploits in <i>ExploitDB</i> .....	42
2.9	<i>CVE-2021-1529</i> as an example of published vulnerabilities in <i>NVD</i> .....	43
2.10	Vulnerability lifecycle adapted from Frei et al. (2006) .....	44
4.1	Research methodology adapted from the research onion model of Saunders, Lewis, and Thornhill (2009a) .....	65
4.2	Design science research process .....	71
4.3	Design science research framework adapted from Hevner and Chatterjee (2010) and Peffers et al. (2007).....	71
5.1	Example of vulnerability data collection .....	88
5.2	Daily published vulnerabilities on NVD from 1999 to November 3, 2021 ...	90
5.3	<i>CVSS</i> version 2 score and metric value distribution for <i>NVD</i> entries .....	90
5.4	<i>CVSS</i> version 3 score and metric value distribution for <i>NVD</i> entries .....	91

5.5	Correlation between working field and usage of <i>CVE</i> and <i>CVSS</i> .....	95
5.6	Potential time delay of scoring and inconsistent scores .....	97
5.7	Overview of the data inconsistency analysis process .....	98
5.8	Structure of the investigated IT and OT systems.....	100
5.9	Data inconsistencies in exploitability and impact between <i>NVD</i> and Vendor. Exploitability and impact analysis are based on the assigned <i>CVSS</i> labels. .	104
5.10	Data inconsistencies in <i>CVSS V2</i> scores between <i>NVD</i> and Vendor. ....	105
6.1	Bird view of vulnerability database overall structure .....	114
6.2	Common data model: source conceptual schema. ....	115
6.3	Data correlation using <i>CVE-ID</i> as the common tag.....	116
6.4	Data correlation using <i>CWE-ID(s)</i> and <i>CAPEC-ID(s)</i> as common tags ..	117
6.5	Example of data correlation for vulnerability <i>CVE-2021-36745</i> .....	117
6.6	Data consolidation structure .....	120
6.7	Common data model: data warehouse conceptual schema.....	121
6.8	Generate query tags .....	122
6.9	Vulnerability-database architecture and query processing .....	124
6.10	Vulnerability reports from synchronized cross-linked database .....	125
6.11	<i>CVSS V2</i> base-score distribution of CI vulnerabilities (2000-2021).....	126
6.12	Threat types targeting CI assets (2000-2021) .....	127
7.1	Vulnerability analysis method using ensemble ML.....	138
7.2	Example of data preprocessing, feature extraction and feature selection ...	140
7.3	High-Level Structure of Vulnerability Severity Computing System .....	146
7.4	Machine learning pipeline based cybersecurity knowledge generation .....	157
7.5	Performance of individual ML models for threat classification .....	164
7.6	Average threat-classification model performance in each training round ...	165
7.7	Best performing model of each training round for threat classification ..	165
7.8	Best performing ML model of each training round for <i>CVSS</i> -characteristic clas- sifications .....	167
7.9	Average model performance in each training round for <i>CVSS</i> -characteristic clas- sifications .....	168
8.1	Connections between taxonomy and the instantiated models.....	174
8.2	Overview of critical infrastructure vulnerability taxonomy .....	175
8.3	Top level cyber-physical taxonomy and instantiation example .....	176
8.4	Cyber, physical and network components in the taxonomy .....	176
8.5	Unidirectional and bidirectional data stream example.....	177
8.6	Taxonomy of Security Objects .....	178

8.7	CI security user interaction model .....	179
8.8	Quantitative vulnerability assessment involving stakeholders .....	180
8.9	Dependence analysis example of a substation RTU .....	181
8.10	Dependence analysis example of manufacturing system .....	182
8.11	Instantiated power-grid substation .....	183
8.12	Public Internet and other networks.....	184
8.13	Office, engineering and security operating center networks.....	185
8.14	Control center network.....	186
8.15	Substation network .....	187
8.16	Power generation and distribution substations .....	188
8.17	Process data and control commands stream .....	189
8.18	Historical analysis and load-prediction data streams .....	189
8.19	Time-synchronization data stream .....	190
8.20	System updating data stream.....	190
8.21	Remote login data streams .....	191
8.22	Web browsing and mail data streams .....	191
8.23	Instantiated <i>Model II</i> based on IEEE 9Bus (with SCADA DMZ zone) .....	192
8.24	Historical analysis and load-prediction data streams in <i>Model II</i> (with SCADA DMZ zone) .....	193
8.25	Cascade failure analysis in <i>Model I</i> and <i>Model II</i> .....	195
9.1	Orchestration of the proposed artifacts and instantiations .....	202
9.2	Connection between the proposed taxonomy and other artifacts .....	203
9.3	A scenario of vulnerability chain in manufacturing system.....	203
9.4	Modeling and identification of vulnerability chain .....	204
9.5	Manufacturing Network.....	205
9.6	Human robot collaboration system that integrates <i>SYMBIO-TIC</i> .....	207
10.1	An overview of relationships between artifacts and research questions .....	213
10.2	Correlation between working field and usage of CVE .....	224
10.3	Correlation between working field and usage of CVSS .....	225
10.4	Cyber and control layers of <i>Model I</i> used in <i>Study X</i> .....	234
10.5	Physical layer of <i>Model I</i> used in <i>Study X</i> .....	235
10.6	Cyber and control layers of <i>Model II</i> used in <i>Study X</i> .....	236
10.7	Physical layer of <i>Model II</i> used in <i>Study X</i> .....	237



# LIST OF TABLES

1.1	Author's contribution level .....	16
2.1	Major differences between ICS and traditional IT networks summarized from Bhamare et al. (2020) and Asghar, Hu, and Zeadally (2019).....	31
2.2	Example of collected information for vulnerability <i>CVE-2021-36745</i> .....	35
2.3	Example of <i>CVE-ID</i> mapping to <i>ATT&amp;CK</i> .....	39
3.1	Text-mining approaches in cybersecurity analysis .....	55
4.1	Design science research paradigm adapted from Saunders, Lewis, and Thornhill (2009b) and Vijay, Bill, and Stacie (2015) .....	66
4.2	Adopted research methods .....	69
4.3	Research process 1 - exploratory studies .....	72
4.4	Research process 2 - vulnerability data curation.....	73
4.5	Research process 3: bridging the data gap.....	75
4.6	Research process 4: vulnerability analysis for critical infrastructures .....	77
5.1	Summary of vulnerability data sources .....	87
5.2	Example of mappingss from vulnerabilities to <i>CWE</i> and <i>CAPEC</i> .....	92
5.3	Example of diverse <i>CWE-IDs</i> allocated to vulnerability instances.....	93
5.4	Deployed vulnerability data sources in <i>Study II</i> .....	100
5.5	Retrieved vulnerability instances in <i>Study II</i> .....	101
5.6	Data inconsistencies in affected products and version ranges between <i>NVD</i> and Vendors .....	102
5.7	Data inconsistencies in the assigned <i>CWE</i> between <i>NVD</i> and Vendor .....	106
6.1	Retrieved additional information for vulnerability <i>CVE-2021-36745</i> .....	118
6.2	CI vulnerability measurement distribution of CVSS V2 base metrics .....	126
6.3	Performance trade-offs between vulnerability data sources .....	130
7.1	Data features, sources, types and modeling .....	141
7.2	Adopted ML model classification performance metrics.....	144

7.3	Performance of CVSS-metric categorization TF-IDF LR ML model .....	150
7.4	CI vulnerability characteristics using <i>CVSS</i> version 3 base metrics .....	152
7.5	Relocating the <i>CWE</i> -IDs for <i>NVD</i> vulnerability entries .....	153
7.6	Remapping of some <i>CWE</i> -IDs .....	154
7.7	List of <i>CWE</i> -IDs for weakness categorization ML training labels.....	155
7.8	Example of the generated <i>CWE</i> experiment corpus .....	155
7.9	Performance of <i>CWE</i> categorization LSTM ML model.....	156
7.10	Evaluation of <i>CVSS</i> V2 score prediction on untrained dataset .....	169
9.1	Vulnerability patch decision making considering criticality and severity....	208

# ACRONYMS

**AI** Artificial intelligence.

**APT** Advanced Persistent Threat.

**CAD** Computer Aided Design.

**CAM** Computer Aided Manufacturing.

**CAPEC** Common Attack Pattern Enumeration and Classification.

**CDM** Common Data Model.

**CERT** Computer Emergency Response Team.

**CI** Critical Infrastructure.

**CISA** Cybersecurity Infrastructure Security Agency.

**CNC** Computer Numerical Controller.

**CPE** Common Platform Enumeration.

**CPPS** Cyber Physical Production System.

**CPS** Cyber Physical System.

**CSRF** Cross-Site Request Forgery.

**CTI** Cyber Threat Intelligence.

**CVE** Common Vulnerabilities and Exposures.

**CVRF** Common Vulnerability Reporting Framework.

**CVSS** Common Vulnerability Scoring System.

**CWE** Common Weakness Enumeration.

**DER** Distributed Energy Resource.

**DoS** Denial-of-Service.

**DSR** Design Science Research.

**FTP** File Transfer Protocol.

**HMI** Human Machine Interface.

**HRC** Human-Robot Collaboration.

- ICS** Industrial Control System.
- ICT** Information and Communication Technology.
- IoT** Internet of Things.
- IT** Information Technology.
- KNN** K-Nearest Neighbor.
- LAN** Local Area Network.
- LDA** Latent Dirichlet Allocation.
- LR** Logistic Regression.
- LSTM** Long Short-Term Memory.
- MiTM** Man in the Middle.
- ML** Machine Learning.
- MLP** Multi Layer Perceptron.
- MSRC** Microsoft Security Response Center.
- MTU** Master Terminal Unit.
- NB** Naive Bayes.
- NC** Numerical Controller.
- NIST** National Institute of Standards and Technologies.
- NLP** Natural Language Processing.
- NN** Neural Network.
- NVD** National Vulnerability Database.
- OSVDB** Open Source Vulnerability Database.
- OT** Operational Technology.
- PLC** Programmable Logic Controller.
- PMI** Product and Manufacturing Information.
- PoC** Proof of Concept.
- RDP** Remote Desktop Protocol.
- RNN** Recurrent Neural Network.
- RTU** Remote Terminal Unit.
- SCADA** Supervisory Control and Data Acquisition.
- SIEM** Security Information and Event Management.

**SOC** Security Operation Center.

**SoS** System Of System.

**STS** Socio-Technical System.

**SVM** Support Vector Machine.

**TARA** Threat Analysis and Risk Assessment.

**TCP** Transmission Control Protocol.

**TF-IDF** Term Frequency-Inverse Document Frequency.

**UML** Unified Modeling Language.

**VND** Vulnerability Notes Database.

**VPN** Virtual Private Network.

**WAN** Wide Area Network.







# CHAPTER 1

## INTRODUCTION

This chapter starts by providing a motivation for this thesis and describes the problem being addressed. Subsequently, the aims and research questions of this research are derived. Next, the main claim driving the thesis investigation is revealed, which is followed by a set of related original contributions. Finally, the outline of this thesis is presented, along with a summary of each chapter content.

### 1.1 CRITICAL INFRASTRUCTURE CYBERSECURITY

Critical infrastructures (CIs), such as energy distribution, manufacturing systems, and transportation roadways, normally combine Information Technology (IT) and Operational Technology (OT) systems (Cardenas et al., 2009). According to the *Critical Infrastructure Sectors* (2022) defined by the United States Cybersecurity & Infrastructure Security Agency (CISA) and the European Program for Critical Infrastructure Protection (Lindström and Olsson, 2009) (Krassnig, 2011), these CIs are essential for maintaining society. For example, the critical manufacturing sector generates products necessary to the other sectors such as electrical grids.

Meanwhile, the rapid advances in information and communication technology (ICT) enable seamless integration of software and hardware. This integration enables a shift from diverse systems empowered mainly by either hardware or software to systems empowered by cyber-physical systems (CPSs) driving emergent systems including Industry 4.0 evolution (Lee, Bagheri, and Kao, 2015) (Xu et al., 2018a). CIs use complex CPSs that link a plethora of physical components from many different vendors to the software systems that control them, in particular, OT systems such as the supervisory control and data acquisition (SCADA) systems. These technological advances have led to the digitization of physical objects and their cyberspace connectivity, to alleviate challenging ultra-dependable critical system services involved in mission-critical applications (Khaiyan and McCalley, 2015), including energy, manufacturing, transportation, healthcare, environmental control, and smart cities (Cardenas et al., 2009). These CIs normally have stringent requirements in terms of timing, reliability and operational resilience (Bordel et al., 2017).

However, the above trend of IT and OT convergence exposes relatively isolated OT equipment to the risks common in IT security protection. Alongside the expected enhancement in efficiency and reliability, the induced connectivity prompted by ICT and its application in SCADA systems expose these CIs to cyber-attacks where conventional security approaches are limited by the scale of the CIs (He and Yan, 2016). Some well-known attacks demonstrate these threats to CIs, specifically targeting industrial control systems (ICSs) (Cárdenas et al., 2011), like the Stuxnet worm (Falliere, Murchu, and Chien, 2011) and the "WannaCry" ransomware (Mohurle and Patil, 2017). Stuxnet was first encountered in 2009 and did not raise wide discussions until 2010. It is the first sophisticated cyber threat targeting ICSs. In 2017, the "WannaCry" ransomware attack occurred

across several manufacturing plants and caused production to stop, incurring substantial business losses. The Ukraine power grid attack in 2015 (Beach-Westmoreland, Styczynski, and Stables, 2016) is another known attack against the power grid system. Liska (2019) summarizes some more recent large-scale ransomware. Furthermore, traditional IT attack methods such as credential theft and Denial of Service (DoS) are proving to be just as effective on OT networks (Bhamare et al., 2020). Attackers often start on the IT network and use IT assets as jump servers to move to more critical OT assets. In July 2020, the news reported a cyber-attack on control systems at water facilities in Israel. Several ransomware such as *EKANS Ransomware and ICS Operations* (2020) encrypt vital data managed by IT (e.g., data historian) and OT (e.g., human-machine interface, or HMI). These attacks exploit vulnerabilities such as feature bypass and improper neutralization, which lead to the production to halt in several automotive factories across Europe, which incurred substantial economical losses (Santangelo, Colacino, and Marchetti, 2021).

CISA summarizes the *2021 Top Routinely Exploited Vulnerabilities* (2022) such as the *Log4Shell* and *Proxyshell* with the corresponding reports disclosed in *Common Vulnerability Enumeration (CVE)* (2022). *Log4Shell* affected the open-source Apache's Java-based *Log4j* logging library, which is widely used in websites, consumer applications, enterprise services, and OT products. It is evident in *CVE-2021-44228* (2021), *CVE-2021-45046* (2021) and *CVE-2021-45105* (2021) and may lead to remote code execution or server compromise once successfully exploited. Although identification and mitigation of affected products using *Log4j* is vital and challenging, software harboring the *Log4j* vulnerability may distribute across the system, making it hard to track all the vulnerable applications, especially the ones provided by third-party vendors. *Proxyshell* refers to a collection of vulnerabilities lying in the client access service of *Microsoft Exchange* server, such as the privilege elevation vulnerability (seen in *CVE-2021-34523* (2021)).

Cybersecurity deals with preventing such threats by identifying related vulnerabilities. Chang (2012) states that cybersecurity involves humans who defend machines and other humans who use machines for attacking purposes. Therefore, cybersecurity definitions are variable and can be influenced by the employed discipline. This thesis adopts the following definition:

*“Cybersecurity is the organization and collection of resources, processes, and structures used to protect cyberspace and cyberspace-enabled systems from occurrences that misalign de jure from de facto property rights.”*

— Craigen, Diakun-Thibault, and Purse (2014)

This thesis uses security and cybersecurity in an interchangeable manner. More specifically, this research focuses on CI network security, in terms of vulnerability assessment to prevent the evolution of vulnerabilities into serious threats. This trend is evolving as a critical and global need to augment existing capabilities of vulnerability assessment instruments. The goal is to withstand the growing and dynamic threat landscape, and to support operational-security managers and executive planners in their efforts to triage the vulnerabilities that present the greatest risks to critical services assurance (Knapp and Langill, 2014).

Vulnerability can be defined in several ways. This thesis considers the perspectives provided by:

*“Vulnerability is a weakness of an asset or control that can be exploited by a threat.”*  
*— ISO/IEC27000:2009 Standardization*

*“Vulnerability is the degree a system is affected by a risk source or agent; or the degree a system can withstand specific loads; or the risk conditional on the occurrence of a risk source/agent; or the uncertainty about and severity of the consequences, given the occurrence of a risk source.”*

*— Society of Risk Analysis*

*“The possible occurrences of vulnerabilities in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source.”*

*— National Institute of Standards and Technologies (NIST) SP 800-160*

Based on these definitions, combining multiple attributes, such as the *criticality* of vulnerable components, the *severity* of emerging *vulnerability* instances, as well as the *likelihood* of exploiting attacks contribute to the computation of the *vulnerability index*. The *criticality* property is weighed with attributes like the applicable product, version range, and functionality importance of the vulnerable component. The *severity* property is measured using standard scoring frameworks such as *Common Vulnerability Scoring System (CVSS)* (2022) or sub CVSS metrics like confidentiality impact. The *likelihood* property is estimated with attributes like attack vector, attack complexity, exploit development status, and remediation development status.

In a CI system, divergent vulnerabilities emerge intermittently in different software, hardware, and firmware. Various buildups of software, hardware and firmware induce vulnerabilities at variant levels of CI structures. Different collections of software products, whether proprietary or commercial, may also have dissimilar vulnerabilities (Kröger and Zio, 2011). Therefore, the vulnerability index of components is affected by the deployment scenario of that component. However, defensive actions against each attempted attack and patching all related vulnerabilities with equal attention would be time consuming and costly. For example, the cyber-threats looming over a power-grid range from malware infections by some proof-of-concept (PoC) attacks to reconnaissance attempts by state-funded hacking groups which could result in launching cyber-attacks with severe consequences (Knapp and Langill, 2014). Therefore, vulnerabilities with higher exploit probabilities or more severe consequences need to be prioritized with corresponding remediation strategies.

Assessing vulnerabilities is supported by data analytics-based decision-making processes to protect CIs and to focus on imminent risks rising from threat exploitability with varying degrees of impact severity. Understanding and measuring vulnerability properties of such networked structures are challenging, yet vital for cybersecurity purposes. A concrete model should be based on evidence from multiple sources of data (Välja et al., 2018). However, the published data is massive with a great level of heterogeneity which needs to be transformed into a common semantic representation (Rahm and Bernstein, 2001) to facilitate machine-readable processes, in order to improve situation awareness applications. Hence, an approach to classify vulnerabilities across components enables an online collection of relevant data to assess vulnerability properties in CIs, as well as to adopt proper defense mechanisms.

Improving the cybersecurity of CIs for both IT and OT networks is vital. However, OT security, especially the security of CPS-based field devices, is overlooked by cybersecurity

rity professionals partially due to the “air-gaped” operation isolation of previous OT devices (Murray, Johnstone, and Valli, 2017). CIs have security challenges different from those found in traditional IT systems due to CI-network complexity and embedded assets heterogeneity (Giraldo et al., 2017)(Zio, 2016)(Cardenas et al., 2009). Zio (2016) points out the gap in how CIs address cyber threats: “*The problem is that the classical methods of system vulnerability and risk analysis cannot capture the (structural and dynamic) complexities of CI; the analysis of these systems cannot be carried out with classical methods of system decomposition and logic modeling*”. Vulnerability-instance response mechanisms in these complex systems are also faced with challenges bridging the knowledge gap between cybersecurity techniques, ICT expertise, and socio-technical management procedures that involve human actors in the production lifecycle.

Next, I state the problem and discuss the related research challenges addressed in this thesis.

## 1.2 PROBLEM DESCRIPTION

The notion of risk remains elusive, as evidenced by the increasing investigations on security operations centers (SOCs) where analysts employ various detection, assessment, and defense mechanisms to monitor security events (Sundaramurthy et al., 2015) (Feng, Wu, and Liu, 2017). Jacobs, Arnab, and Irwin (2013) suggest several ways to build a SOC. Still, on average SOC can quickly generate several GBs of security events per day, which can create a significant stress on human responders (Bhatt, Manadhata, and Zomlot, 2014).

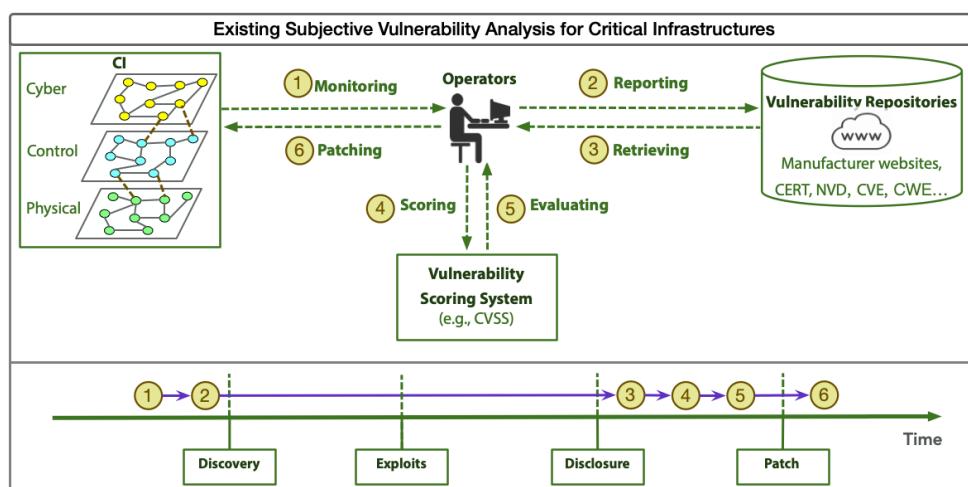


Figure 1.1: Current subjective vulnerability analysis for critical infrastructure

To manage vulnerability, security operators in SOCs need to monitor their system by keeping pace with vulnerability repositories, as presented in Figure 1.1. Typically, SOCs involve multiple security tools (e.g., network vulnerability scanners) combined with analysis of data contained and produced by CI operations as well as alerts retrieved from vulnerability repositories such as *CVE* (Russo et al., 2019). The operators need to follow *CVSS* to evaluate the severity of each identified vulnerability (Agyepong et al., 2019). In

addition, security operators need to further forecast the match between these vulnerabilities and the intricate layer networks of CIIs to prioritize and guide patching exercises (Hafiz and Fang, 2016). This process is illustrated in Figure 1.1, which shows the central role of security operators in SOCs and their need for support to keep pace with the evolving vulnerability alert repositories.

Although existing security mechanisms illustrated in Figure 1.1 attempt to standardize severity schemes to build a common awareness about the criticality level of vulnerabilities across CI networks, there are still crucial research gaps to address. Below, I summarize the four main challenging research issues: (i) heterogeneous and diverse vulnerability data sources; (ii) subjective and human-centered process; (iii) dependencies exacerbate CI vulnerability analysis; (iv) gaps in vulnerability management.

#### *(i) Heterogeneous and diverse vulnerability data sources*

A strong community effort has been established to improve sharing, standardization, and automation of cybersecurity issues in the form of universal vulnerability databases such as CVE and the accompanying cybersecurity terminologies. The *CVE* database is one of the most influential forces in sharing and standardizing vulnerability information among the cybersecurity community efforts. *CVE* repository discloses more than 160 000 reported and exploited vulnerabilities between 1999 and October 2021. It provides standardized but unstructured textual descriptions of vulnerabilities, which are fed into the US-based *National Vulnerability Database (NVD)* (2022) for further vulnerability assessment such as vulnerable system configurations using *Common Platform Enumeration (CPE)* (2022), weakness categories using *Common Weakness Enumeration (CWE)* (2022), and severity scoring using *CVSS*. Vulnerability-mitigation decisions that rely on *CVE* or *NVD* records as primary data sources can be biased (Christey and Martin, 2013) by ignoring other sources of data (Anwar et al., 2020)(Jo et al., 2020). Some third party analyzers such as *Computer Emergency Response Team* (2022) (CERT) and manufacturer websites such as *Microsoft Security Response Center (MSRC)* (2022) provide valuable insights into cybersecurity from the industrial perspective. In addition, NVD, CISA, and MSRC provide contradicting analysis of reported vulnerabilities (Anwar et al., 2020). Vulnerability assessment in CIIs requires analysis of heterogeneous data sources to discover correlations between dependent cybersecurity dimensions, such as vulnerability, attack, and threat instances. These instances are stored as standalone traces in separated enumeration lists, which adds to the challenges of data correlation.

#### *(ii) Subjective and human-centered process*

The increasingly reported vulnerabilities in public repositories and the potential to exploit them pose severe challenges to the cybersecurity research community (Croft, Xie, and Babar, 2022). According to the *CVE* report, since 1999, more than 170,000 vulnerabilities have been reported (last checked March 13, 2022). For example, DoS threats have been triggered around 27,800 times using vulnerability exploits (last checked on *CVE Details* (2022) March 13, 2022). However, many vulnerabilities are not reported in *CVE* or are newly emerging, which exacerbates the evaluation process. Public vulnerability databases rely heavily on manual reporting such as the *CVE Numbering Authorities* (2022) led process and further analysis, which leaves room for potential errors (Tang, Alazab, and Luo, 2017). Existing vulnerability analysis approaches such as *CVSS* calculator (Joh and Malaiya, 2011) require subjective and manual input, based on qualitative judgments of vulnerability properties such as exploitability, scope and impacts. Users need to manually select inputs for each vulnerability-metric, which requires expert knowledge. This tedious and time-consuming investigation process can take hours, days, weeks, or even more to complete when conducted manually (Le, Chen,

and Babar, 2021). Moreover, possible delays in risk remediation due to manual assessment can increase risk exposure. Therefore, relying on the knowledge of individual experts only could introduce recurrent costs, subjective evaluations and contradicting outcomes (Chan et al., 2019). Nevertheless, security operators of CI also need to obey limited budget restrictions, and consider the limited computing resources of CI networks.

*(iii) Dependencies exacerbate CI vulnerability analysis*

Current cybersecurity efforts mostly deal with singular vulnerabilities that occur in isolation. However, groups of vulnerabilities need to be assessed simultaneously as the result of combining multiple vulnerabilities (Humayed et al., 2017). This is challenging since CI incorporates complex data from multiple interconnected physical and computation assets (Ouyang, 2014). Different buildups of software, hardware and firmware induce vulnerabilities at various levels. However, due to the interconnections among systems, vulnerabilities emerge constantly at a pace faster than human involvement can assess their severity (Ashibani and Mahmoud, 2017). This prompts a threat-agent to exploit a sequential chain of vulnerabilities to trigger advanced persistent threats (APTs) (Chen, Desmet, and Huygens, 2014), whereby exploits on one component may give access to another component that can be exploited. Such chained vulnerabilities need to be identified and assessed, to diagnose system-level vulnerabilities and potential threats at an earlier stage of possible attacks. Of particular importance are the identification and analysis of explicit dependencies that cross CIs.

*(iv) Gaps in vulnerability management*

The disparity of terminology used in cybersecurity and CI domains exacerbates correlation analysis and knowledge extraction pertaining to CI cybersecurity (Barnaghi et al., 2012) (Mozzaquattro et al., 2018). Dynamic and complex processes within CIs involve multi-domain enterprise management procedures, which may result in communication gaps throughout interconnected application-specific sub-systems of the overall production fabric (Wu et al., 2018) (Lee, Bagheri, and Kao, 2015). This is exemplified by different priorities in IT and OT cybersecurity practices. That is, IT security usually focuses on the confidentiality, integrity, and availability of critical data, whereas OT security focuses on the protection of production loss or safety (Conklin, 2016). Limited collaboration between different departments of IT and OT also contributes to the knowledge gaps in cybersecurity assessment of CIs (Vielberth et al., 2020). Moreover, it is challenging to extract and manage system configuration information from CIs (Bernstein and Haas, 2008). Normally, operators need to query different PCs/machines following various vendors' suggestions. For example, one may obtain embedded software in a Windows computer by using PowerShell (Shepard, 2015). However, vendors use various semantics and syntaxes, which increases the difficulty of information integration. Moreover, many critical infrastructure companies outsource their IT or OT services to other companies, which further enlarges the knowledge gap between different sub-systems (Kandias et al., 2011).

To summarize, the imprecise nature of vulnerability assessment and the huge volume of scanned data increase the burden for security analyzers. Vulnerability assessment is both time-consuming and prone to errors when conducted manually.

In addition, advancements in security-focused data-driven solutions shift the burden of large volume vulnerability management from security experts and to the digital alternatives (Chan et al., 2019). The thorough literature review by Le, Chen, and Babar (2021) has identified a trend of combining the strengths of artificial intelligence (AI) tools such as machine learning (ML) techniques and human intelligence to proactively detect and analyze threats and to provide actionable insights to security analysts for making in-

formed decision. This raise questions about how to deliver a holistic data-driven security strategy that helps security management (e.g., operators and executives) evaluate risks arising from various vulnerability sources and their combination across CI networks.

The next section presents the research aims and objectives, and how they address the above challenges and stated problems.

## 1.3 RESEARCH AIM AND OBJECTIVES

Rather than delivering a solution with an all-encompassing data format for security information and connectors that correlates all popular formats across existing systems, this thesis contributes to the understanding and development of vulnerability analytic solutions for complex CIs, particularly in the context of diverse, heterogeneous and complementary sources of cybersecurity incident reports. This is formally described by the following aim:

*Aim:* To investigate workflows that bridge the knowledge gaps among different security functions (e.g., vulnerability management, report analysis, and infrastructure networks monitoring) to correlate vulnerability findings and coordinate mitigation responses in complex CIs.

The proposed research framework can be used to help companies address the above-mentioned cybersecurity challenges and facilitate the adoption of data driven techniques in related vulnerability analysis. Hevner and Chatterjee (2010) make the following claim: “*Phenomenon is typically a set of behaviors of some entity that is found interesting by the researcher or by a group – a research community. Understanding is knowledge that allows prediction of the behavior of some aspects of the phenomenon.*” The results of this thesis should help cybersecurity stakeholders gain *understanding* (“*descriptive advance*”) of CI cybersecurity phenomena such as vulnerability properties and CI dependencies, *managing* (“*prescriptive advance*”) of vulnerability repositories and CI system configurations, as well as *forecasting* (“*predictive advance*”) of vulnerability trends in CI environment (see sub-section 1.7.2 for further discussions on descriptive, prescriptive and predictive advances).

The following research objectives are identified to achieve the aim of this thesis:

*Objectives:*

- (A) To expand the scope of knowledge about vulnerability alerts and curate meaningful context of vulnerability analytical processes.
- (B) To assess identified vulnerabilities with enhanced levels of automation that reduce existing information gaps induced by ad-hoc and subjective auditing processes.
- (C) To model vulnerability in CIs in a reproducible manner that supports vulnerability assessment and increases the level of security awareness.

## 1.4 RESEARCH QUESTIONS

To address the above-mentioned challenges and research problem, four research questions were formulated to guide the investigation. The proposed research questions support future security operation systems in complex CIs to augment the competence of security teams with AI-rooted cybersecurity techniques. These questions emphasize issues that alleviate the burden of dealing with a large volume of cyber-security data and the drudgery of manually managing their versatile instances as a part of auditing exercises. In addition, the formulated questions are designed to improve situation awareness and support cooperative security management.

The above-mentioned rationale that drives the investigation is explored via the following four interrelated research questions (RQs), all of which are centered on the vulnerability assessment process for CIs, as illustrated in Figure 1.2. These four RQs correlate in a way that allows one RQ to answer another RQ. Each RQ addresses the previously stated research objectives.

*Research Questions:*

- RQ (1) What are the challenges in critical infrastructure vulnerability assessment using publicly accessible vulnerability repositories?
- RQ (2) How can data be obtained and correlated for vulnerability analysis considering complex and heterogeneous sources of vulnerability alerts?
- RQ (3) How can the missing information gap in the curated and correlated vulnerability database be bridged?
- RQ (4) How can the vulnerabilities of complex critical infrastructures be modeled and assessed with the support of a curated database and vulnerability assessment algorithms?

This thesis has resulted in six research publications and two that are currently under review. Each paper answers one or more of the RQ(s), as illustrated in Figure 1.2. These RQs are explained next to show how they address the challenges mentioned in subsection 1.2.

*RQ (1) - What are the challenges in critical infrastructure vulnerability assessment using publicly accessible vulnerability repositories?*

This thesis focuses on how to make cybersecurity deductions and decisions in an automated way using data-driven techniques, while addressing the following problems. First, data-driven solutions use AI techniques such as ML algorithms. These systems learn by training and incorporating additional data as they iteratively refine their analysis. Thus, the availability, precision, and conflict-free data sources are paramount, considering security perspectives. In addition, security experts still have a critical role to dedicate their intuition, creativity and experience as fact-checkers and champions of these solutions (Veksler et al., 2018). Therefore, it is important to explore the challenges of using publicly accessible cybersecurity repositories for CI vulnerability assessment, especially from a security stakeholder's perspective.

*RQ (2) - How can data be obtained and correlated for vulnerability analysis considering complex and heterogeneous sources of vulnerability alerts?*

Vulnerability analysis uses heterogeneous data sources to discover correlations be-

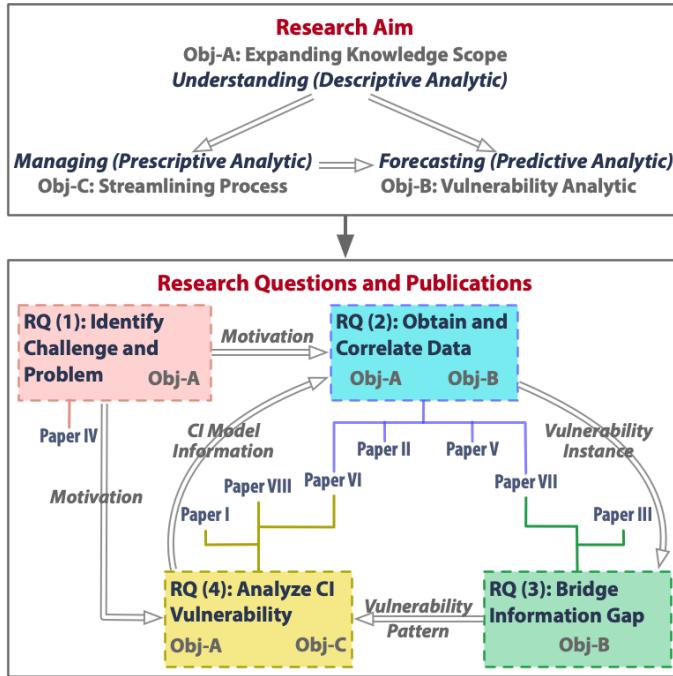


Figure 1.2: Connections between research aims, questions and resulted papers

tween dependent cybersecurity dimensions, such as vulnerability, attack, and threat, which are collected as standalone traces. This information appears in separate repositories, and may be published in different data formats. Furthermore, these repositories use proprietary standards involving their own syntax and semantics. Therefore, answering this question requires integrating multiple vulnerability data sources. When answering this question, research challenge (i) (heterogeneous and diverse vulnerability data sources) is considered. Namely, vulnerability data sources are heterogeneous, incomplete and redundant. Cross-referencing these data pools followed by a thorough analysis of inter-spectral and intra-spectral features could provide an effective classification of vulnerabilities (Bullough et al., 2017).

*RQ (3) - How can the missing information gap in the curated and correlated vulnerability database be bridged?*

As pointed out in RQ (2), the challenges of vulnerability analysis are due to heterogeneous vulnerability data sources as well as their incompleteness. To bridge such information gaps as well as to generate intuitive vulnerability patterns, this research question also addresses research challenge (ii) (subjective and human-centered process), following a human-centered process. A large number of published vulnerabilities continue to grow. Meanwhile, third-party components incorporating various software embedded in CI fabrics, with possible new vulnerabilities, will continue to be developed. Therefore, it is crucial to identify vulnerability patterns quickly, from multivariate time-varying data, each with thousands of possible observation instances.

*RQ (4) - How can the vulnerabilities of complex critical infrastructures be modeled and assessed with the support of a curated database and vulnerability assessment algo-*

rithms?

This research question is formulated in response to the research challenges (iii) (dependencies exacerbate CI vulnerability), which refers to the need for dependence identification in CI vulnerability and (iv) (gaps in vulnerability management), which refers to the communication gap in interpreting security indicators in vulnerability management. When a vulnerability is exploited in a CI asset, the threat agent might open a backdoor or gain system authority to compromise other assets in the system. Some unexpected and unforeseen vulnerabilities could be exploited to trigger severe attacks across the entire CI fabric. Hence, it is necessary to model how chained-vulnerabilities are exploited step-by-step across the CI fabric and their potential impact to derive a CI-level vulnerability score. To do so, the basic structure and functions of CI networks need to be abstracted to exhibit desired properties like functional dependence, which addresses challenge (iii). In addition, a common framework provides a unified understanding from different views and prevents potential threats to infiltrate through cross-layer gaps and reduce the magnitude of their impact, which addresses challenge (iv).

## 1.5 THESIS STATEMENT

This thesis answers the proposed research questions. The proposed architecture of data-driven vulnerability analysis orchestrates a range of modules (see Figure 1.3).

The proposed indicators of **data-driven vulnerability** require human operators to comprehensively assess cyber vulnerabilities in deployed CI networks such as power grid and manufacturing systems. The goal is to reduce security management gaps induced by ad-hoc and subjective auditing processes as well as to increase the level of automation in vulnerability analysis. More specifically, this proposed approach suggests constant monitoring and maximum use of information to elicit vulnerabilities and their severities before damage is done. Information about vulnerabilities is reported in natural language expressions across numerous, and multifaceted repositories, as well as hundreds of manufacturer websites and thousands of security blogs posted every day (Lun et al., 2016). Data-driven security employs ML techniques to “understand” and take advantage of this increasingly massive corpus of information and to unfold their hidden correlations (Andrade et al., 2019). In doing so, data-driven security empowers SOC operators at various organizational hierarchy levels with the ability to detect changes in their scope of activity and analyze these changes with as much context as possible to distinguish and eliminate new threats.

Moreover, Figure 1.3 also highlights four important artifacts of this thesis, as a result of achieving the stated objectives in section 1.3:

### **Artifact I - Vulnerability data model**

This artifact includes a conceptual common data model (CDM) for cybersecurity data source integration, which defines a source data mapping schema and the related data warehouse structure, as well as a query generation method to support vulnerability instances retrieval with CPE tags. The proposed model is instantiated into a cross-linked database from 12 widely used vulnerability-alert repositories and standardized enumerations. This artifact is described further with details such as correlation algorithm cross-link processes in Chapter 6.

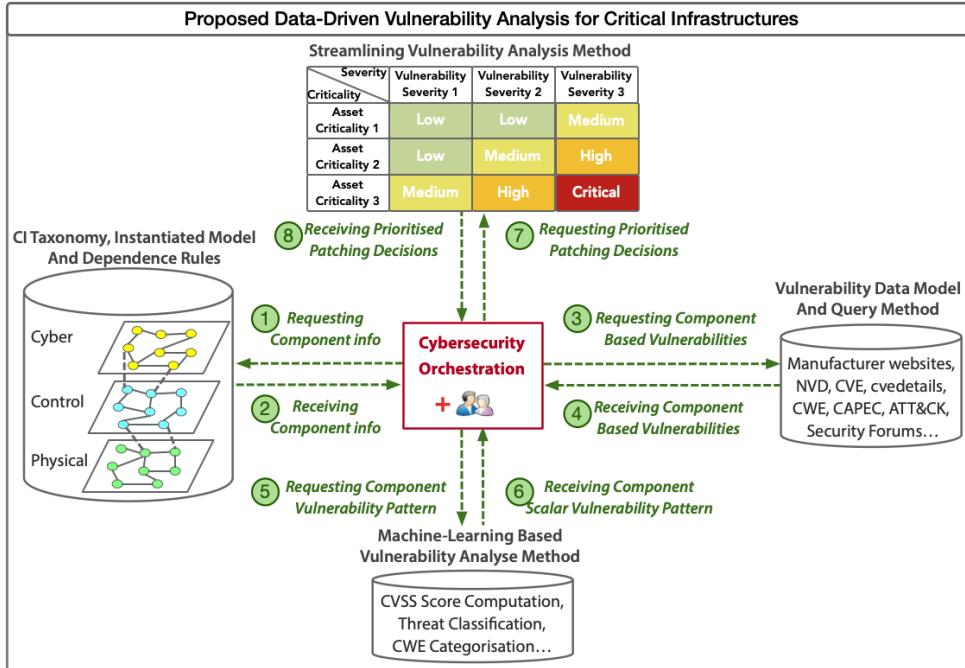


Figure 1.3: Proposed data-driven vulnerability analysis for critical infrastructure

#### Artifact II - ML-based vulnerability analysis method

This artifact transfers retrieved vulnerability instances from **Artifact I** into a set of rule-based classifiers to correlate vulnerability patterns to fill in the information gaps, and to compute a corresponding vulnerability index. This artifact is described in Chapter 7 and is further composed of the following approaches:

- **Artifact II-A:** A ML approach that automatically assigns CVSS-metric based labels to textual vulnerability reports. This approach can be customized to accommodate a preferred CVSS version to allow a common computational semantic that improves consistency in vulnerability assessment. This approach produces better performance than similar published CVSS computing methods.
- **Artifact II-B:** A ML approach that pinpoints the CWE weaknesses as root causes of disclosed vulnerabilities at a higher abstraction level.
- **Artifact II-C:** An approach that utilizes an Ensemble algorithm to achieve better performance at classifying benign connections compared to linear ML models. This overarching method features an optimization opportunity to discover the best model for different vulnerability analysis tasks.

#### Artifact III - CI vulnerability taxonomy and dependence rules

This artifact addresses CI semantics, such as components' topological connections, functional processes and security properties, through a taxonomy (**Artifact III-A**). On top of the taxonomy, cyber and cyber-physical functional dependence rules (**Artifact**

**III-B**) are defined to support cascade modeling and the evaluation of vulnerability combination to deliver a holistic asset-level or even system-level vulnerability assessments that are meaningful across the hierarchy of organization roles. This artifact is instantiated into power grid and manufacturing networks as **Artifact III-C** for validation, with more details provided in Chapter 8 and Chapter 9.

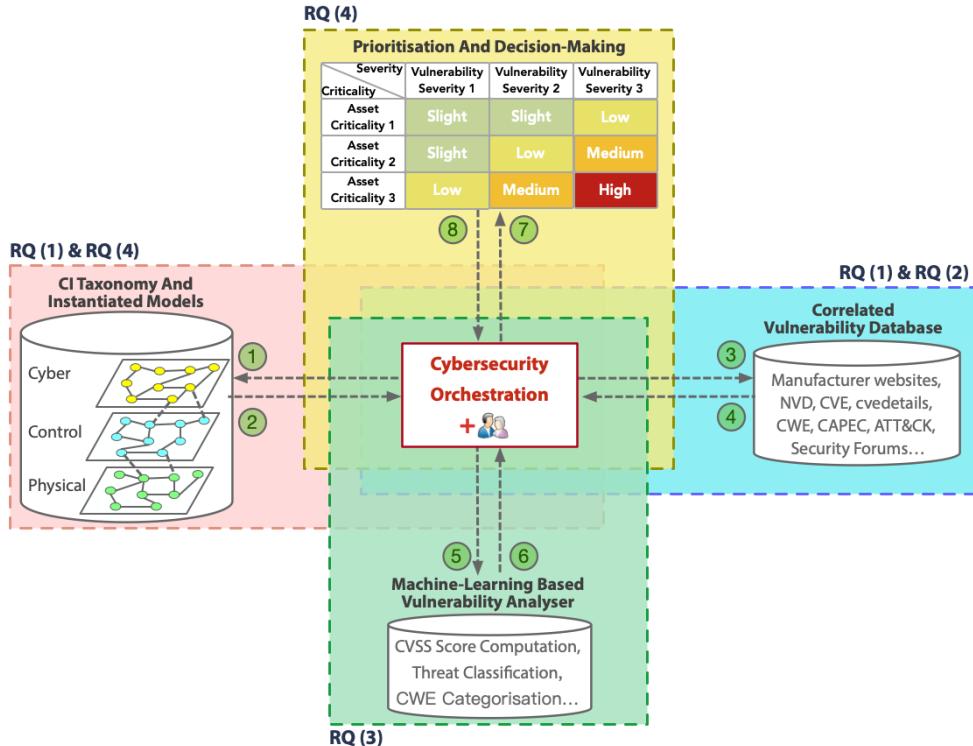


Figure 1.4: Connections between research questions

#### Artifact IV - Vulnerability analysis streamlining method

This artifact synthesizes a vulnerability index based on the severity generated in **Artifact II** and dependence index produced in **Artifact III** to support prioritization of mitigation mechanisms such as patching exercises, with more details in Chapter 9.

Furthermore, the research questions discussed in Section 1.4 can be mapped to the proposed cybersecurity architecture in Figure 1.3, following the illustration in Figure 1.4. *Cybersecurity orchestration* is the central coordinator agent that facilitates communication and exchange of information between the proposed modules and orchestrates their activities.

More specifically, in Step 1 and Step 2, RQ (1) and RQ (4) related workflows contribute a domain-specific language to elicit cyber (IT), cyber-physical (OT) and physical components and their relationships in infrastructure repositories, which serves as a knowledge base of CI models to support vulnerability analysis performed in subsequent steps of the framework. In Step 3 and Step 4, RQ (1) and RQ (2) workflows contribute an approach whereby heterogeneous data is collected periodically from cybersecurity repositories,

which are then correlated to standard enumerations of cybersecurity related categorizations, and are further integrated into a common localized database. Another contribution is the correlation techniques from retrieved data from this database to discover vulnerability instances and their severity. In Step 5 and Step 6, RQ (3) workflows contribute text-classification models that are leveraged to produce relevant patterns of threat, vulnerability and weakness to support risk management exercises, based on data retrieved from the constructed localized database. In Step 7 and Step 8, RQ (4) workflows empower security operators with indicators that prioritize vulnerability assessment based on vulnerability severity as well as component criticality. This process allows vulnerability management at the operational, management and executive level to evaluate and reflect on machine-generated indexes to feedback the central cybersecurity orchestration module. The outcomes from these steps lead to indicators that support the know-how of cybersecurity decision-makers (see Figure 1.5).

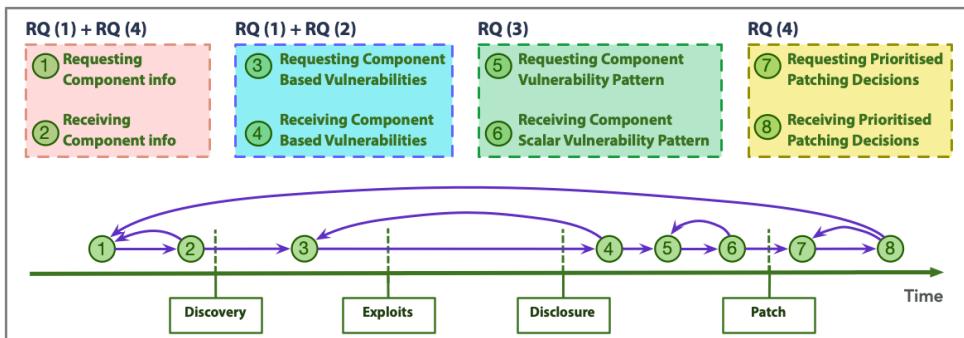


Figure 1.5: Work flows in the proposed framework

## 1.6 CONTRIBUTIONS

The main contribution of this thesis is the elaboration of novel data-driven vulnerability analysis solutions for complex CIs. This section discusses five perspectives around this main contribution: identifying problems and challenges; expanding vulnerability knowledge scope; bridging the information gap; benchmarking CI vulnerability; and streamlining vulnerability analysis.

### 1.6.1 IDENTIFYING PROBLEMS AND CHALLENGES

We reveal the current use of various cybersecurity standards, particularly interpretation use-cases of these enumerations and their correlations with other data sources. We also conducted an in-depth analysis of historical vulnerability records in *NVD* features such as weakness types, severity and exploit threat. We carried out a survey of CI cybersecurity participants' sentiments on open-source cybersecurity repositories and CVSS in practice. Based on this survey study, we selected the metrics for evaluating or designing data-intensive vulnerability assessment/management solutions for CIs. We further performed a case study on vulnerability assessment using software and hardware configurations of an actual large data center. We used *CVE*, *NVD* and vendor websites as sources for vulnerability identification and retrieval. We conducted follow-up interviews of three cybersecurity specialists in the investigated infrastructures, to collect their feedback on

the employed study results as well as their insights on publicly available cybersecurity data sources at large.

In doing so, we make the following contributions to the field:

- Identifying heterogeneity, incompleteness and inconsistencies of vulnerability reports.
- Motivating vulnerability data curation and severity scoring automation.

### 1.6.2 EXPANDING VULNERABILITY KNOWLEDGE SCOPE

*Artifact I* enables context-aware data analysis that aids situation awareness with up-to-date vulnerability trends, expanding the vulnerability knowledge scope. The proposed data model correlates repositories of vulnerability reports such as *NVD* and *Shodan Database* (2022) and standardized enumerations like *CWE*, allowing further discovery of inconsistent labels assigned to *CVE* vulnerability instances. For example, considering reconciled vulnerability scores as the ground-truth, severity scores of newly reported vulnerabilities are computed using *Artifact II* considering various CVSS metric labels. *Artifact I* also includes a query generation module that supports further vulnerability retrieval, while addressing the synonyms issues in vulnerability repositories. In addition, the instantiated database of *Artifact I* is synchronized with multiple publicly accessible repositories of vulnerability reports to narrow further the risk-window induced by discovered vulnerabilities.

### 1.6.3 BRIDGING THE INFORMATION GAP

*Artifact II* is composed of a set of ML methods that infer missing security information such as exploit threat and weakness category in reported vulnerability instances, facilitating a greater level of automation of vulnerability assessment. More specifically, two ML approaches infer separately CVSS severity scores and CWE weakness categories of reported vulnerability instances. Similarly, the ensemble-based ML method was developed to combine independent classifiers that use information from different sources, yielding better performances for cybersecurity classification tasks. This approach embodies a complete ML pipeline which includes modules for data pre-processing and cleaning, feature engineering, model selection, and ensemble-model construction built on top of trained models to optimize the classification of security indicators. Optimization opportunities are released at each step through the process to find the best ensemble model for different tasks.

### 1.6.4 BENCHMARKING CI VULNERABILITY

*Artifact III* delivers a modeling methodology of intricate networks and related constraints of CIs in terms of vulnerability analysis. The modeled common semantics of both IT and OT entities support IT/OT security convergence and provide a unified understanding from different views, which prevents potential threats from infiltrating through cross-layer gaps and reduces the magnitude of their impact. Furthermore, the dependence rules derive the functional dependence structure from data flow specifications and can be used to pinpoint the most critical components in a CI model. This extends works on traceability of enterprise models by linking IT components to OT entities and to the components of the physical layer such as the electrical grid.

Multiple extensive and realistic reference models are instantiated to define aspects of power-grid and manufacturing systems such as the control center, substations, and the data/control flows between software components. These instantiated models can also serve as a base that is analyzed by external tools for vulnerability analysis. Static analysis queries are used to pinpoint dependencies and weaknesses in the CI layered network. In doing so, the proposed taxonomy and implemented queries provide a system architecture-aware prioritizing analysis.

### 1.6.5 STREAMLINING VULNERABILITY ANALYSIS

A CI asset can be subject to multiple vulnerabilities given two facts: one asset is composed of several components; and one component may have several vulnerabilities. A systematic computational approach to system-wide vulnerability needs to combine all the relevant measurements, considering their dependency rules. *Artifact IV* contributes to such systematic CI vulnerability analysis while streamlining the assessment processes. The function of this artifact is to streamline the interplay between vulnerability data and CI layered system configuration to facilitate a greater level of automation in vulnerability assessment. For example, *Artifact IV* ranks prioritization of vulnerability patching by taking into consideration vulnerability severity (from *Artifact II*) and component criticality (from *Artifact III*), to further optimize security investments and resource allocation to reduce the potential risk window. The vulnerability analysis orchestration approach assists system administrators in identifying the most crucial components for cybersecurity protection, considering that different degrees of patch priority are required for components with the same vulnerability because some components are protected or less accessible (e.g., an isolated device).

The next sub-section clarifies the author contributions in the included publications.

### 1.6.6 AUTHOR CONTRIBUTIONS

Author's level of contribution is further summarized in Table 1.1 where *Major* implies that work carried out mainly by the main author, *Medium* implies work carried out mainly through a cooperation between the main author and co-authors, and *Minor* implies little involvement from the main author.

## 1.7 RESEARCH SCOPE

Cybersecurity is inherently an interdisciplinary field. Chang (2012) highlights in one of the Guest Editor's Column of The Next Wave that in addition to the critical traditional fields of computer science, perspectives from other relevant fields (e.g., electrical engineering and mathematics) are needed to deliver a multidisciplinary approach.

### 1.7.1 KEY RESEARCH ELEMENTS

The research conducted and presented in this thesis involves four key research elements (see Figure 1.6): cybersecurity (the research subject content), CI (the domain of application), data (the basis of the proposed solutions), and people (the user of the proposed solution tools).

Table 1.1: Author's contribution level

Paper	Idea	Metric	Model	Data	Analysis	Writing
Publication I	Minor	Minor	Major	Major	Medium	Medium
Publication II	Major	Major	Major	Major	Major	Major
Publication III	Major	Major	Major	Major	Major	Major
Publication IV	Medium	Major	Major	Major	Major	Major
Publication V	Medium	Major	Major	Major	Major	Major
Publication VI	Major	Major	Major	Major	Major	Major
Publication VII	Major	Major	Major	Major	Major	Major
Publication VIII	Major	Major	Major	Medium	Major	Major

"Idea" and "Metric" refer to research idea and research metrics formalization, separately. "Model" includes model conceptualization, design and development. "Data" is about data obtaining and cleaning.

The cybersecurity research scope in this thesis is mostly relevant to the definition given by Craigen, Diakun-Thibault, and Purse (2014) presented in sub-section 1.1. In the context of this research, "*cyberspace*" refers to traditional information technology infrastructures such as telecommunication networks and computer systems (NIST, 2013), and "*cyber-space enabled systems*" refer to cyber-physical systems. The proposed AI-based approach used to enhance the capability of cybersecurity in the context of this thesis is mapped to the definition fragment referring to "*collection of resources, processes, and structures*". People working in "*the organization*" also match the proposed human-in-the-loop approach to cybersecurity.

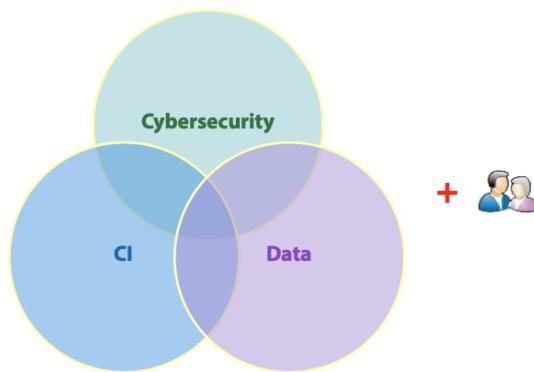


Figure 1.6: Key research areas and elements

Although technical perspectives have been the dominating factor of discourse in cybersecurity research, SoC actors have a vital role in making use of the data-driven technologies that are supported by cybersecurity management. Socio-technical systems (STS) proposed originally by Trist (1981) in the 1940s were suggested to address the broader issue of cybersecurity. In their definition, technology (covering machines, embedded technologies, and the associated processes), as well as society (covering working groups and organizations), need to be given equal weight. In the scope of this thesis, cybersecurity in CIs is perceived as a STS problem, involving both technical and social dimensions

of cybersecurity.

### 1.7.2 RESEARCH AREA

Xu (2019) proposes the concept of *cybersecurity dynamics*, to capture the complexity of cybersecurity and to reflect the awareness of the global cybersecurity state of a system. A separated fundamental concept such as confidentiality, integrity and availability issues subsequently lead to the rise of cryptography, encryption, intrusion detection, and similar isolated solutions (Xu, 2014). Instead, *cybersecurity dynamics* views these building-block mechanisms from a holistic perspective. As originally stated by Lin, Lu, and Xu (2019), a cybersecurity-dynamics model characterizes a system's state under a particular set of attacks from the attacker's perspective as well as a particular set of defenses from the defender's perspective. The state of awareness derived from the knowledge of real-time or near real-time global cybersecurity indicators empowers cyber defense decision-makers with highly effective defense postures (Chen, Cho, and Xu, 2018). According to Xu (2019), the primary goal of modeling cybersecurity dynamics is to gain *descriptive*, *predictive*, and *prescriptive* empowerment. Descriptive power refers to the capability to quantitatively characterize a system state under different attack and defense scenarios (Delen and Demirkhan, 2013). Predictive power is about the ability to forecast cyber threats to support proactive defense mechanisms (Larose, 2015)(Hafiz and Fang, 2016). Descriptive and predictive powers provide a prescriptive view to the state of awareness, suggesting cyber defense activities involving systems that combine actionable data and user feedback (Lepenioti et al., 2020)(Sukhija et al., 2019).

This thesis focuses on vulnerability assessment for CI, which is aligned with this new concept of *cybersecurity dynamics*. That is, the efforts are concentrated on three perspectives of the defender:

- First, a common semantic structure is employed to support modeling approaches to security applications. This proposed semantic framework represents connections between CI configurations and vulnerability categories to support the identification of security gaps (Syed et al., 2016). A vulnerability can be modeled as a property of a CI component. Dependence rules and quantitative metrics are then added to support queries such as vulnerability severity (i.e., achieved through exploit scenario and cascade modeling) to gain the *descriptive* capability (Hendrickx et al., 2014) (Teixeira et al., 2015).
- Second, data analytic techniques are adopted to characterize CI and vulnerability model parameters to gain predictive capability to forecast vulnerability trends. Analyzing vulnerability trends employs online cybersecurity data sources as input into *predictive* analysis to evaluate vulnerability severity and occurrence likelihood based on some computational intelligence techniques (Bullough et al., 2017) (Tavabi et al., 2018) (Almukaynizi et al., 2017) and (Xu et al., 2018b).
- Third, combining the power of computational intelligence for cybersecurity with human perception enhances the *prescriptive* ability for situation awareness and efficient mitigation reaction when detecting vulnerabilities.

Figure 1.7 illustrates the above-mentioned disciplines along with a three-dimensional space that forms the scope of the proposed multidisciplinary research area. We use knowledge and approaches from these research disciplines to produce CI taxonomy,

vulnerability patterns, as well as system-level vulnerability models, all of which serve as essential components in our data-driven vulnerability assessment solution.

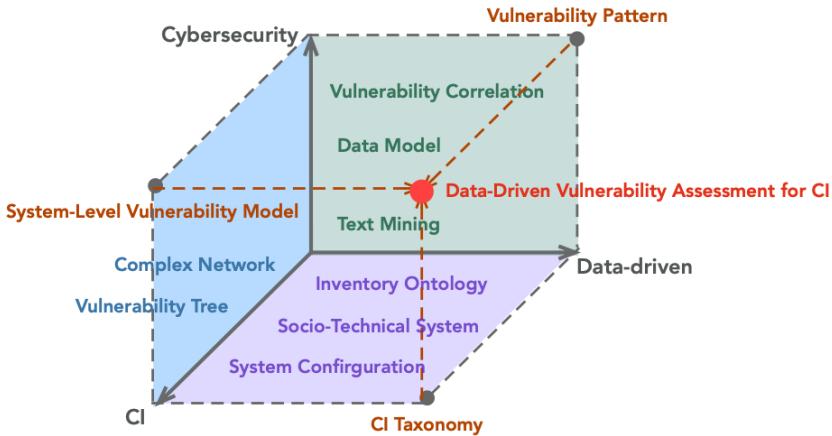


Figure 1.7: Related research disciplines

### 1.7.3 RESEARCH FOCUS AND DELIMITATION

Cybersecurity research in CI is a multidisciplinary field. The focus of this thesis is on vulnerability-driven risk analysis for CI. The definition of risk adopted is:

*“A measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of: (i) the adverse impacts that would arise if the circumstance or event occurs; and (ii) the likelihood of occurrence.”*

— NIST SP 800-30 and NIST Cybersecurity Framework Version 1.1

Risk assessment studies may focus on different cybersecurity perspectives. In this interdisciplinary field, some works concentrate on modeling attack steps or attack-vectors (Johnson et al., 2016c). In contrast, other works address the vulnerable nature of the system while trying to root out vulnerabilities used by exploit-vectors (Zio, 2016). Both types of works are used to model cyber-threat patterns and support cyber-risk assessment (Mozzaquattro et al., 2018), which can be seen as a bow-tie relationship as suggested by Ciapessoni et al. (2016).

Shevchenko et al. (2018) summarize commonly used threat analysis methods, such as Threat agent and risk analysis (TARA) and Microsoft’s STRIDE threat model, which identifies and rates potential threats to determine which threats to mitigate first and which countermeasures to take. TARA uses three major libraries based on incident reports and security measures collected and developed by *Intel Security* experts, i.e., threat agent library, methods and objectives library and common exposure library (Rosenquist, 2009). Rosenstatter and Olovsson (2018) use TARA to identify threat exposures and security strategy alignment in automotive system, but at a high-level overview. STRIDE evaluates the system detail design, and is usually used to analyze specific threats in the design phase of software or hardware, particularly spoofing, tampering, repudiation, information disclosure, DoS and elevation of privileges (Shostack, 2014). Another threat assessment and remediation analysis method uses web-based tools to search and process catalog-stored attack vector and to map countermeasure data (Wynn, 2014).

Shevchenko et al. (2018) compare the trade-offs between multiple threat modeling methods, and suggest that CVSS has the advantage in terms of automated components and built-in prioritization of threat mitigation.

Starting with vulnerability identification, this thesis assesses risk of CIs to analyze the likelihood if a vulnerability will be exploited and impact of an exploited vulnerability. That is, the proposed approaches could help to identify components containing new vulnerabilities in a fast and automated manner, with the support of publicly accessible vulnerability repositories. There are also some other requirements such as attack detection and resilience analysis (Uday and Marais, 2015). However, these alternative cybersecurity approaches are outside the scope of this thesis. It should be noted though, if extended with aforementioned TARA and STRIDE threat libraries, the proposed approaches can also support threat identification and assessment.

The primary scientific approach of this thesis lies in the discipline of informatics, so the focus is also on computational and socio-technical aspects. The computational approach is oriented to the data-driven techniques and simulation methods adopted in this thesis. The socio-technical approach is related to the organizational impact of the presented approach that involves human actors in the loop.

Due to privacy and security concerns, security experiments in real systems are limited and might expose these systems to potential threats. Instead, computer simulation approaches are adopted in this thesis together with an in-depth literature review and interviews with industrial practitioners, to enable the applicability of the model. This approach also captures the critical properties of a system's structure and its dynamics. For example, interviews with CI security professionals enable the collection of functional and topological information of industrial production processes in the context of cybersecurity. Based on the collected information, CI networks are modeled to reflect real infrastructure connections, to support vulnerability-centered simulations with reliable predictions.

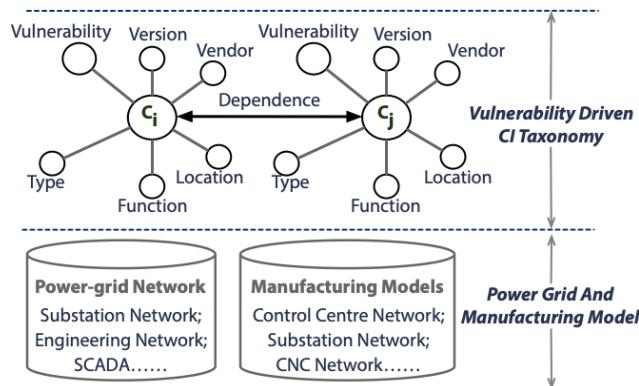


Figure 1.8: Scope of CI reference model and instantiation

The herein proposed approaches are mainly demonstrated through studies in the CI sectors of energy and critical manufacturing (see Figure 1.8). The proposed CI taxonomy contains both components and their interconnections in terms of vulnerability analysis. The instantiated model includes detailed models for power-grids and manufacturing networks. Therefore, the CI taxonomy and instantiated model play an important explanatory role together. In this thesis, only the dependencies within an infrastructure

are modeled. However, possible dependencies between different infrastructures are out of the scope of this thesis. These instantiated models also work as a base to store model specification, component connections, as well as vulnerability properties, which enables model transformation from a high-level abstraction to artifacts that mimic real-life infrastructures.

It should however be emphasized that the problems identified and the methods proposed in this thesis are not exclusively related to power grid and manufacturing systems, and are not limited to applications in CI cybersecurity protection. In fact, the advocated data-driven vulnerability analysis methods are applicable to IT and OT systems as well as their convergences with variant scales from individual IoT devices to complex automotive driving systems, for example.

## 1.8 THESIS STRUCTURE

This document is composed of ten chapters. A summary of the content of each chapter is provided below.

Chapter 1 provides a preliminary introduction to the research area around cybersecurity in CIs. This introduction is followed by a presentation of some research challenges in vulnerability assessment methods faced in complex CIs. Secondly, the burden of dealing with an extensive volume of vulnerability data and existing manual auditing processes are discussed. The proposed thesis statement is revealed, which is followed by delimitation of scope to clearly define the research boundaries. This chapter concludes with a description of the thesis structure and a summary of each of the subsequent chapters.

Chapter 2 presents relevant background and concepts. CI and CPS definitions are provided with emphasis on cybersecurity issues. This chapter discusses three semantically interconnected CI concepts (component, asset and system) as well as the vulnerability lifecycle and related characteristics. This chapter also introduces relevant cybersecurity repositories and enumerations used in this thesis, including the existing CVSS standard.

Chapter 3 presents a theoretical background in our research area. Related works are discussed to highlight similarities and differences compared to the proposed approaches.

In Chapter 4, the principal research methodology guided by design science principles is presented, which is followed by a discussion of applied research methods during our research process. Some evaluation methods and metrics used across the research methods during the research process are also discussed.

The main research results are presented in Chapter 5, Chapter 6, Chapter 7, Chapter 8, and Chapter 9. These results are used to discuss the modules making-up the proposed data-driven cybersecurity solutions illustrated in Figure 1.3.

In Chapter 5, exploratory studies and related RQ(1) results are presented to introduce identified challenges in using publicly accessible cybersecurity repositories in CI vulnerability assessment from security stakeholders' perspectives. These exploratory studies include a baseline study, a questionnaire-based survey investigation, and a case study on vulnerability assessment using real system information of an actual large data center.

In Chapter 6, the proposed vulnerability data model and related RQ(2) results are presented. A cross-linked and correlated database to collect, extract, and visualize vulnerability data across multiple existing repositories is revealed. The database is then used in a study for in-depth vulnerability data analysis centered on CI components, to explore the trends of CI vulnerability.

In Chapter 7, results illustrating the proposed ML methods and related RQ(3) results are presented. This chapter outlines ML techniques to select optimal predictive models for different vulnerability-analysis tasks such as threat-, weakness- and severity-related classifications. An evaluation benchmark for the proposed ensemble approach is shown to validate the performance of the proposed approach.

In Chapter 8, the results about the proposed CI taxonomy and partial RQ(4) results are presented. A domain-specific language and its instantiated models are illustrated to represent CI functional dependencies and topological structures. A study related to a power grid infrastructure and corresponding IT/OT network is also presented in this chapter.

In Chapter 9, the domain-specific language proposed in Chapter 8 is further integrated into a vulnerability analysis streamlining method that enhances the collaboration between different actors throughout CI-based infrastructure. A study in a manufacturing infrastructure environment is presented to illustrate the application of the proposed method.

In Chapter 10, some concluding remarks are provided along with a discussion of some future research directions. More specifically, this chapter revisits the research questions and objectives proposed in Chapter 1 to present contributions and limitations.



## KEY CONCEPTS AND BACKGROUND KNOWLEDGE



# CHAPTER 2

## KEY CONCEPTS AND BACKGROUND KNOWLEDGE

This chapter starts with brief introductions of the research area and the most relevant disciplines, namely cyber-physical layered critical infrastructures, cybersecurity, and related data-driven analysis. Four key research elements (CI, cybersecurity, data, and stakeholders) are discussed that characterize the cybersecurity study in this thesis. A conceptual framework is then provided to illustrate some used terminologies in this thesis. In doing so, some important concepts such as vulnerability, exploit, and risk are defined. CI structure as well as underlying component, asset and subsystem concepts are introduced. The focus of this thesis is on vulnerability analysis and thus related concepts such as vulnerability lifecycle and vulnerability characteristics are discussed.

### 2.1 CI CYBERSECURITY CONCEPTS

According to Burgess (2010), cybersecurity research is intrinsically about factors, principles, and circumstances that form a conceptual background so that key actors subjectively come to a shared understanding and an agreement on how to respond to security threats. Furthermore, Spring, Moore, and Pym (2017) highlight that standard definitions are necessary as a common language or an ontology of terms to distinguish information-security science from information-security practices. Therefore, this section provides some conceptual definitions of commonly used cybersecurity-related terminology throughout this thesis.

The grounding of concepts serves as a key basis that correlates multiple disciplines and can be implemented through ontologies (Martins et al., 2020) (Mavroeidis and Bromander, 2017). An ontology is a structured form of knowledge presentation that specifies concepts and their relationships, and can integrate information coming from different sources. We construct a conceptual model on top of a vulnerability management ontology developed by Wang and Guo (2009), as illustrated in Figure 2.1. This figure presents a concept map of cybersecurity terms and their semantic relationships.

In the conceptual framework, *vulnerability* is a significant cyber-security issue hampering the protection of critical infrastructures (Humayed et al., 2017). *Risk* is broadly defined as the potential occurrence of events or incidents that might materially harm the system, as introduced earlier in 1.7.3. *Risk* assessment considers two main factors, namely the likelihood and the impact of vulnerability exploitation that targets a particular CI *component*, to obtain malicious access, or cause *damage* to operational functions (Humayed et al., 2017) (Goerlandt and Reniers, 2016). Conceptually, infrastructure *operators* need to gauge investments against risk indicators using specific benchmarks. These indicators rank assets by their importance, their vulnerability, and the consequence of threats that exploit those vulnerabilities (Kure, Islam, and Razzaque, 2018). The reported *mitigation vectors* represent an expert-system interpretation of numerical risk-indicators to assist operators in carrying out patching decisions (Ruan, 2017). The

mitigation vector is a translation of numerical risk-indicators into operational instructions that support operators in the course of patching action.

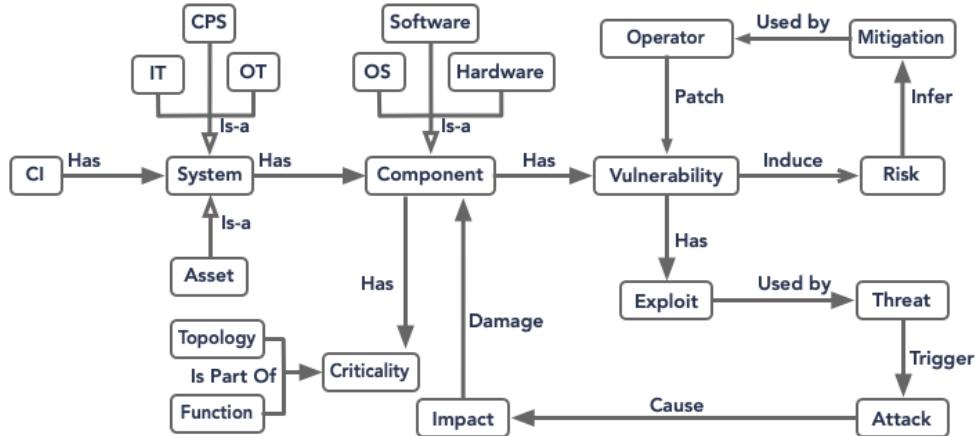


Figure 2.1: Critical infrastructure cybersecurity concepts

In addition, a vulnerability assessment examines the possibility of asset disruption (Ani, He, and Tiwari, 2017) (Kröger and Zio, 2011). However, not all assets have the same criticality qualities, referred to as criticality vectors in the following text. To prioritize vulnerability assessment activities, it is necessary to identify critical nodes based on their *criticality-vectors*. These are the nodes whose disruption would have the most repercussions. A system with particular criticality is regarded as an *asset*. Vulnerabilities may be exploited by a *threat-agent* in different ways using *exploit-vectors*. A threat agent may further trigger an attack in different ways using *attack-vectors*. Each attack vector represents a specific attack scenario which may result in different implications, represented by *impact-vectors*. Each impact-vector may be used to represent the consequence of a disruption to a targeted asset in CIs in different ways. The vulnerability-assessment process mainly involves the identification, quantification (Ezell, 2007) and evaluation of how exploit-vectors trigger attacks and the magnitude of those attacks represented by impact-vectors.

### 2.1.1 COMPONENT, ASSET AND SYSTEM

This thesis adopts the definitions of *system* and *system element*, or *component* as:

*“System is a set of interacting elements that are organized to achieve one or more stated purposes.*

*Asset fulfils specified requirements and may be implemented via hardware, software, or firmware; physical structures or devices; or people, processes, policies, and procedures.”*

— ISO/IEC/IEEE 15288:2015 Standard

Some researchers use the definition of *device* instead of asset (Griffor et al., 2017). This

this thesis regards an asset made up of components and with particular value to stakeholders:

*“An item of value to stakeholders. An asset may be tangible (e.g., a physical item such as hardware, firmware, computing platform, network device, or other technology component) or intangible (e.g., humans, data, information, software, capability, function, service, trademark, copyright, patent, intellectual property, image, or reputation). The value of an asset is determined by stakeholders in consideration of loss concerns across the entire system life cycle. Such concerns include but are not limited to business or mission concerns.”*

— National Institute of Standards and Technologies (NIST) SP 800-160

The connections between a component, an asset, and a system are illustrated in Figure 2.2. A vulnerability could be regarded as an emergent property of a component within CIs. For example, vulnerabilities  $V_{3,2}^1$  and  $V_{3,2}^2$  exist in the component  $C_{3,2}$  that is embedded in the asset  $A_3$ . Moreover, the asset  $A_3$  is part of the system  $S_1$ . In a complex CI, hardware, software, and OS are assembled and used in different ways, which might create various binaries with potential backdoors. As highlighted in common standards like Corporation (2014) and ISA (2007), the first step in securing a CI is identifying the critical assets. Protecting this critical subset of components can enhance the robustness of a system to sustain reliable operations (Vukovic et al., 2012).

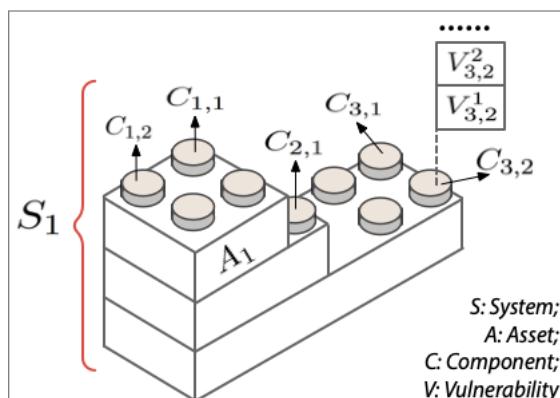


Figure 2.2: System, asset and component

CPS is one type of systems. Lee (2015) regards CPS as only the intersection of the *Physical* and the *Cyber*. In this thesis, CPS is viewed from a broader perspective, i.e., the union of the *Physical*, the *Cyber*, and the *Control* between the former two. Considering the nature of CPS, we define a CPS as composed of software (e.g., firmware, toolset, software library) and hardware (e.g., a hard drive). Note that software further subsumes OS (e.g., a Windows OS). OS functionally manages software components and acts as an interface between application software and hardware. Software is integrated in hardware and hence relies on this component's power supply and CPU. In the interim, software monitors, regulates, and acts upon hardware components. In addition, CPS requires a reliable network connection to transport data and complete feedback loops. Typically, a CPS asset can be for instance a human machine interface (HMI) or a programmable logic controller (PLC).

### 2.1.2 CRITICAL INFRASTRUCTURE

A complex CI is a system of systems (SoS) that integrates a collection of devices to achieve desired capabilities (Uslar et al., 2019). The dependencies in such a SoS are divided into inter- and intra-dependencies.

The growth and scale of CI complexity are ever-evolving due to the fast expansion of networked applications in *smart* systems, which are overseeing a range of industries and CIs. These systems use a network of embedded sensors, platforms, and actuators to perceive and affect a physical-process that typically requires guaranteed quality-of-service performance provided by safety-critical applications (Lewis, 2019). For example, the smart grid combines multiple electric-power production plants with multiple loads using dynamic load-balancing and dynamic pricing to meet demand-response strategies (Zio and Sansavini, 2013). Manufacturing systems or cyber-physical production systems (CPPSs) combine monitoring components into networked closed-loop systems with humans in the loop to improve production workflows (Lee, Bagheri, and Kao, 2015), which relies on industrial control systems (ICS). Today's automotive systems also employ CPSs into a range of applications, including avionics, railroads, and traffic management. A prominent example of CPS in smart traffic is the control of autonomous vehicles.

ICS is a typical OT architecture in CIs, consisting of automation control components and process control components that gather and monitor real-time data in order to assure the automation, process control, and monitoring of industrial infrastructure. The fundamental ICS components consist of SCADA, PLC, HMI, distributed control system, remote terminal unit (RTU), master terminal unit (MTU), and interface technology that enables communication between these components. A PLC is an essential CPS component that regulates industrial devices to maintain production processes. A RTU transmits to an MTU system telemetry data from sensing devices linked with physical power components. A HMI is either a standalone device or an embedded communication interface for visualizing and monitoring MTU actions and RTU data flow (Humayed et al., 2017)(Corporation, 2014). RTU and MTU are linked to other SCADA components, such as SCADA servers, via routers, fiber optic cables, and switches.

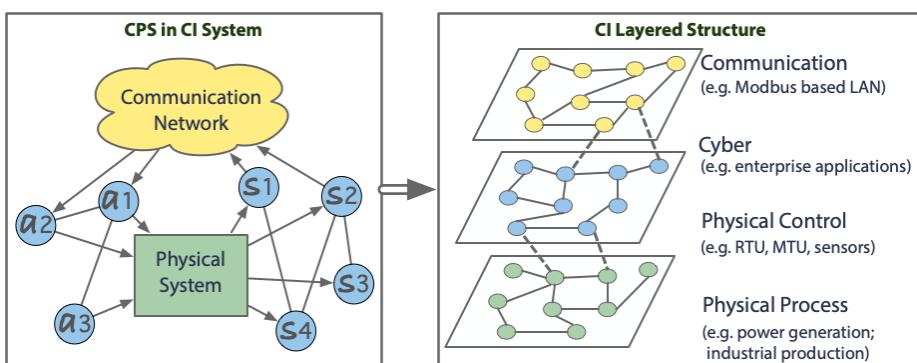


Figure 2.3: Layered architecture of critical infrastructures

The networked structure of CIs integrates interrelated-layers where each layer represents a subsystem of the CI architecture, as illustrated in Figure 2.3. This figure also depicts the interaction between growing CI layers, highlighting the prominent role of CPS such as SCADA and other control systems. These subsystems communicate with one another through data linkages. In the meantime, each layer connects components

by intra-communication data lines. Within and across layers, many communication protocols are used. This layered model covers the functions and topological structure of CI in order to provide insight into the features of CI and forecast its behavior.

The bottom physical-layer models the physical-process dynamics and properties. It features a network of sensors (e.g.,  $s_1$  and  $s_2$ ) and actuators (e.g.,  $a_1$  and  $a_2$ ), embedded into physical-process components, as illustrated in Figure 2.3. The middle control-layer embodies the feedback-logic loop by collecting, monitoring, and estimating states from sensors and correcting errors via manipulation mechanisms of the physical process using actuators. The control layer features a network of OT components to coordinate and synchronize operations. Finally, the top cyber layer expresses decision-support analytic to manage the underlying control-system operations in an enterprise IT-networked platform. The role of this platform is to optimize the efficiency and to monitor the security of the physical process operations. The control center provided by this layer incorporates an IT network of workstations and servers, including application and data-store servers.

### 2.1.3 CYBER-PHYSICAL SYSTEM CHARACTERISTIC

In addition to CPS, there are many associated terminologies such as industrial Internet, Internet of Things (IoT), machine-to-machine, smart cities, and others that describe similar or related systems and concepts (Lee, 2015). This thesis adopts the definition of CPS as:

*“CPS integrates computation, communication, sensing, and actuation with physical systems to fulfill time-sensitive functions with varying degrees of interaction with the environment, including human interaction.”*

— NIST CPS Public Working Group (2017)

This definition describes well the essential characteristics of CPS, which are further discussed next. Considering the CPS framework proposed by NIST CPS Public Working Group (2017) and the CPS definition given by Berkeley CPS Project (2022), the characteristics of a CPS are defined based on the conventions for a system architecture framework given by ISO/IEC/IEEE 42010. This means the herein proposed framework contains interpretation and use of CPS architecture views to frame cybersecurity concerns.

As illustrated further in Figure 2.4, CPSs are feedback systems distinguished by their control loop, which takes the system output into consideration in order to adjust desired response performances. Such feedback systems have mainly four characteristics that distinguish CPS from traditional or pervasive interconnectedness, namely connectedness between the cyber and the physical, heterogeneity and complexity, adaptive and predictive, as well as time-sensitive (Khaitan and McCalley, 2015):

- Connectedness between “*Cyber*” and “*Physical*”: The combination of the cyber and the physical, and their cyber-physical connectedness, is essential to CPS. Practically, CPS involves traditional IT that takes measurements from sensors on physical processes for data analysis, and also involves traditional OT that processes these measurements to drive actuators that affect back the physical processes.

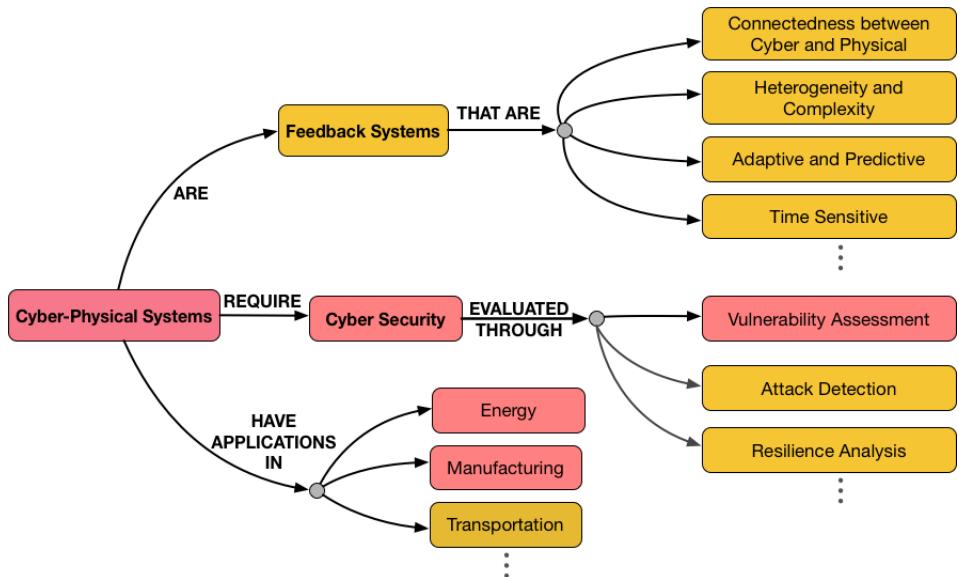


Figure 2.4: Cyber-physical system characteristics summarized from *Cyber-Physical Systems - A Concept Map* (2022)

- **Heterogeneity and Complexity:** CPSs may be developed into a SoS (Eusgeld, Nan, and Dietz, 2011) accommodating a variety of computational models together with variant physical components. In a typical CPS structure, a controller continuously or periodically observes the physical process, and continuously or periodically provides actuation to distributed physical devices, via software and networks. Such networked and/or distributed structure leads to heterogeneity and complexity of CPSs. While behavior analysis in CPSs needs to take into account emergent behaviors or interactions between CPS subsystems.
- **Adaptive and Predictive:** CPS are characterized by their interactions with their operating environment through closed-loop control. These control strategies need to be adaptive to respond to changing conditions, and predictive to anticipate changes in physical processes. CPS intelligent abilities may enable such changing capacities for perceiving, learning and operating.
- **Time-Sensitive:** Finally, cyber-physical systems are typically time-sensitive, with software that has timing constraints, including tasks that must be executed periodically within accurate time intervals. Moreover, given their networked-nature, CPSs leverage network time-synchronization to support distributed actions-coordination.

## 2.1.4 IT SECURITY AND OT SECURITY

Rapid advances in ICT enable seamless software and hardware integration in CIs. However, the growing exposure of CIs to cyberspace for increased efficiency and intelligent controls, lead to increased surfaces that are vulnerable to cyber attacks, which raises security concerns in terms of intrusion and tampering by adversaries (Lun et al., 2016). Table 2.1 compares traditional IT security and OT security (particularly ICS security) from Bhamare et al. (2020) and Asghar, Hu, and Zeadally (2019).

Table 2.1: Major differences between ICS and traditional IT networks summarized from Bhamare et al. (2020) and Asghar, Hu, and Zeadally (2019)

Property	Industrial Control System (ICS) Network	Traditional Information Technology (IT) Network
Network Edge	ICS network edge has intelligent devices containing sensing and control functions.	IT Network edge has server computers or workstations.
Architecture	Vertically integrated structure between the master node and the terminal node.	Rather flat structure with peer-to-peer relationship between servers.
Performance Requirement	Real-time communication with critical response time; Limited delay; Medium throughput.	No specific requirement for real-time response; Can tolerate delay; High throughput.
Availability Requirement	High availability; Any discontinuity needs to be planned in advance.	System restarting is tolerable.
Risk Management	Safety concerns; Fault tolerance of the production process.	Confidentiality and integrity of dataset; Business impact.
Security Maintenance	Thorough testing and incremental deployment are carried out before making changes or update to the system.	Automated software update can be arranged.
Operating System	<i>VxWorks, Linux, Windows</i> , etc.	<i>Windows, Linux, Unix</i> .
Data Exchange Protocol	Open Platform Communications ( <i>OPC</i> ), Distributed Network Protocol 3 ( <i>DNP3</i> ), <i>Modbus</i> , etc.	Transmission Control Protocol ( <i>TCP</i> ), Internet Protocol ( <i>IP</i> ).

Jang-Jaccard and Nepal (2014) propose three categories of vulnerabilities, namely hardware, software, and network infrastructure and protocol vulnerabilities. Hardware derived vulnerabilities are mostly seen in the form of unauthentic or illegal hardware clones. Hardware, software, and OS are assembled and used in different ways within CI fabrics, creating various binaries with potential backdoors (Humayed et al., 2017) (Ashibani and Mahmoud, 2017). One example of hardware vulnerability is no physical-access protection which an attacker might exploit to gain unauthorized physical access. And hence, exploiting hardware base vulnerabilities enables the threat agents to access or alter physical elements of a computer server (e.g., a hard drive) or a network (e.g. a router). Software oriented vulnerabilities exist in system firmware or application software. An outdated software with flaws in source code might be exploited by a bypass threat that is further materialized by a code-injection attack triggered by malicious actors. Network infrastructure and protocol vulnerabilities frequently appear in network protocols such as transmission control protocol (TCP).

Intricate network structures and inherent vulnerability gaps (Rosas-Casals, Valverde, and Solé, 2007) distinguish the problem of vulnerability assessment in complex CIs that utilize CPS, as highlighted by Humayed et al. (2017) that “*the complexity of CPSs and the heterogeneity of CPS components have introduced significant difficulties to security and privacy protection of CPS*”. The major differences between OT (normally CPS-based) security and the traditional IT network security are summarized in Table 2.1. The complex structure of CPSs and the limited computing capability of CPS-based control and sensing devices lower the protection degree to withstand cyber threats targeting dynamically evolving vulnerabilities (Lun et al., 2016). Many industrial CPSs are built using legacy devices with limited computing sources. At the same time, an ICS designed for reliability and longevity can easily be outpaced by the tools employed by an attacker (Rouse, 2003). However, purely adding vulnerability patching into existing CPSs may create back-doors that allow exploits to happen. Increasing attacks were recorded that target directly OT networks, including the Stuxnet worm (Falliere, Murchu, and Chien, 2011) and multiple ransomware based attacks like *Ransomware Impacting Pipeline Op-*

*erations, Alert (AA20-049A) (2020).*

The proliferation of CPSs into CIs such as an energy-distribution system requires trust. Such trust is supported by cybersecurity of CPS networked structures to withstand malicious attacks, prevent intrusions and sustain resilience of physical processes (Nguyen, Ali, and Yue, 2017) (Khaitan and McCalley, 2015) (Bordel et al., 2017). Due to the complexities and challenges unique to these vulnerable networks, there are no easy solutions to CI cybersecurity problems. For example, limited computing resources are available to networked devices, which are embedded in physical processes.

### 2.1.5 POWER GRID SECURITY AND MANUFACTURING SECURITY

This thesis proposes vulnerability analysis approaches that support risk assessment of CIs and are validated in the domains of power-grids and manufacturing systems, as discussed in this section. The power grid is the infrastructure supplying electrical energy with increasing challenges of efficiency and reliability, while energy is paramount to economic development and social welfare. Manufacturing system sets up the foundation for production of goods that help maintaining of the society.

The growth and scale of both power-grid and manufacturing systems' complexity are ever-evolving due to the fast expansion of networked applications. Taking the power-grid system as an example, it is distinguished by the enormous scale and intricate interconnections of the network carrying power flows. The network includes power components tied up together via transmission and distribution lines to form a complex system connecting power-generation sources to power-consuming loads. Power is, however, difficult to store, which requires continuous real-time supply-demand synchronization. The power grid employs CPS in evolution from aging power-delivery systems, in order to optimize and protect electricity delivery operations (Humayed et al., 2017). These processes could be facilitated by the analysis of data that originate from the different layers composing the CPS in power-grid architectures. For instance, the control layer includes a network of microprocessor-controlled physical objects, such as RTUs, which interface with physical process sensors and actuators. On top of the cyber-physical layer, control center applications process these measurements to support operational power-flow decisions to balance the supplied and demanded power flows (Knapp and Langill, 2014).

Manufacturing systems face similar challenges in cybersecurity protection due to the increasing connectivity facilitated by communication links within and across CPS networks, as shown in Figure 2.5. In the physical layer, engineers or operators could locally maintain workstations using local stations or HMIs. In the control layer, OT administrators, engineers or operators could remotely maintain workstations using remote stations or through a virtual private network (VPN), to optimize production operations. Then, engineers use control and command servers to process these data to support operational production decisions, and to synchronize their operations. The top cyber layer expresses decision-support analytics to manage the underlying control system operations, in an enterprise platform of application servers and datastore. The application servers provide various application services, such as computer aided design (CAD) server, software-update server, time-unit server, and web server. The datastore includes process-data server, historian database, and domain controller. Specifically, design engineers use CAD program to design product through application servers and store the corresponding 3D-model files in data-store servers. Software administrators use software-update server to update outdated firmware or system software, with the support from historian-database. The process data server stores and transmits design, process and manufacturing data from production flow, which supports file transfer between data analyzers.

The historian database stores historical data from application servers, which is queried by operators to monitor production processes. The domain controller reserves user-information, and supports corresponding authorization maintained by administrators.

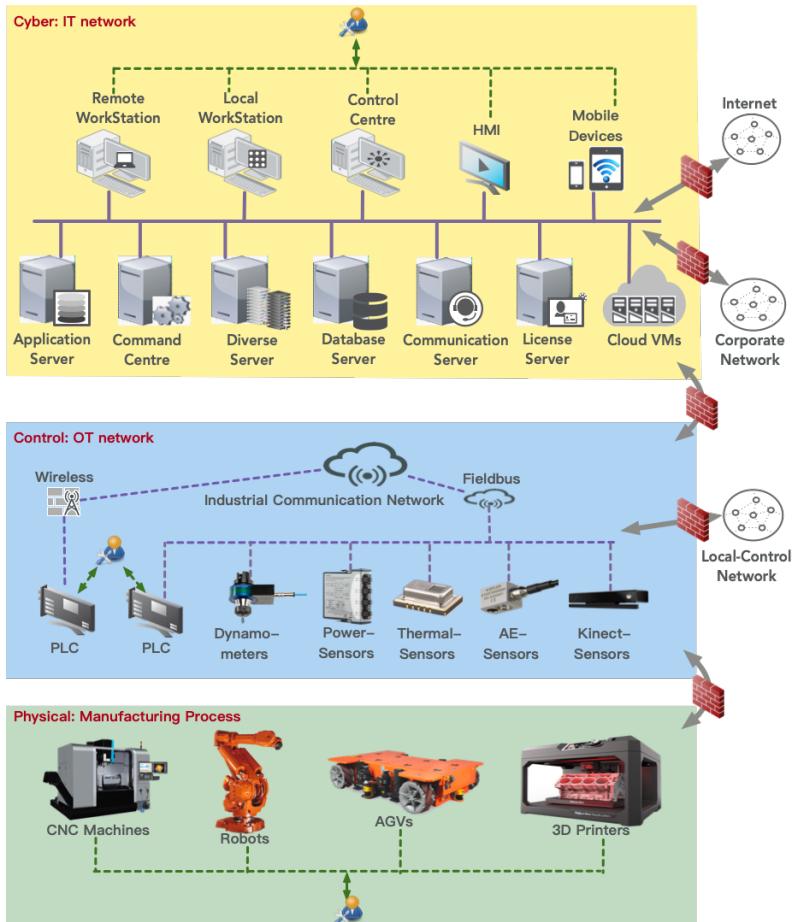


Figure 2.5: Layered structure of manufacturing system

From an abstract level, power grid and manufacturing systems share similar CPS layered structures, as illustrated earlier in Figure 2.3. However, such a new interdependent relationship between different systems also introduce new vulnerabilities to the system. Such dependencies need to be identified and assessed systematically.

Dynamic and complex physical processes involve further multi-domain enterprise management procedures, which may result in communication gaps throughout interconnected application-specific sub-systems of the overall critical infrastructure fabric (Wu et al., 2018). Taking the manufacturing system shown in Figure 2.3 as an example, Numerical controllers (NC) machining part describes a production system typically across four stages, namely part-design that defines its product and manufacturing information (PMI) data, process planning that creates the detailed NC machining process data, part machining that runs this process data on computer numerical controller (CNC) machines and the tool condition data to monitor the production process, as well as quality-

inspection that involves quality-assessment data. Groups of application-specific staff are responsible for design, machining, and inspection activities within the production process, such as designer, process planner, CNC machine operator and quality inspector. Software administrators are mainly responsible for operational and maintenance tasks to ensure capabilities of software-as-a-service within the cloud-based environment, to enable the services of software programs such as CAD and computer-aided manufacturing (CAM) programs. However, in current manufacturing management structures, the communication between different groups is limited due to inherent differences in working contents. A successful attack that propagates without notice could result in severe impact, due to lack of communication through manufacturing networked-layers and related operators such as network administrators, application-specific engineers and security managers. Therefore, a concrete model should be based on multiple sources of heterogeneous data which needs to be transformed into a common semantic representation (Välja et al., 2018), and in a machine readable format, to improve a common view of situation awareness. Our proposed taxonomy supports a greater level of communication between vulnerability-handling stakeholders to enhance such cybersecurity situation awareness.

## 2.2 VULNERABILITY DATA SOURCE TAXONOMIES

This section provides an overview of vulnerability data sources and open taxonomies used in this thesis, which are categorized into scoring mechanism, enumeration, vulnerability repository, and sharing standard. This section analyzes the semantics of six widely adopted cybersecurity enumerations (e.g., *CWE*), and study the accessibility, interpretability and syntax of seven commonly used cybersecurity repositories (e.g., *CVE* and *Shodan*).

Table 2.2 lists the features of vulnerability instance *CVE-2021-36745* that are extracted from the collected *CVE* and *NVD* data sets. The description allocated by *CVE* is used for further analysis by *NVD*, vendors, and third-party analysts. In addition, *NVD* provides vulnerable system configurations in *CPE* metadata format, weakness ID assigned from *CWE*, and also severity scores in *CVSS* V2 and V3, separately.

### 2.2.1 SCORING MECHANISM

*CVSS* is developed and maintained by the *CVSS* Special Interest Group and reported in the Forum for Incident Response and Security Teams. *CVSS* standard scheme is widely used to support quantitative vulnerability-severity assessment in both academic research (Khazaei, Ghasemzadeh, and Derhami, 2016) (Johnson et al., 2016b) (Spanos, Angelis, and Toloudis, 2017) and security-critical industrial domains (Stine et al., 2017). Using *CVSS* mechanisms, a vulnerability instance could be identified by different types of properties measured through inherent metrics following a range of corresponding measurement values (Pendleton et al., 2016). The principal properties of an asset vulnerability are categorized into three groups, namely *Base*, *Temporal*, and *Environmental* properties. The base-property refers to inherent characteristics of a vulnerability that do not change over time or over deployment environments.

We first introduce base score properties of *CVSS* Version 2 or V2, which are grouped into two classifications, namely exploitability  $P_{Exploit}^{v2}$  and impact  $P_{Impact}^{v2}$  properties, which we discuss next.

Table 2.2: Example of collected information for vulnerability CVE-2021-36745

Feature Name	Source	Value
CVE ID	CVE	<a href="#">CVE-2021-36745</a>
CVE Description	CVE	A vulnerability in Trend Micro ServerProtect for Storage 6.0, ServerProtect for EMC Celerra 5.8, ServerProtect for Network Appliance Filers 5.8, and ServerProtect for Microsoft Windows/Novell Netware 5.8 could allow a remote attacker to bypass authentication on affected installations.
CVE Reference	CVE	<a href="https://success.trendmicro.com/jp/solution/ooo289030">https://success.trendmicro.com/jp/solution/ooo289030</a> <a href="https://success.trendmicro.com/solution/ooo289038">https://success.trendmicro.com/solution/ooo289038</a> <a href="https://www.zerodayinitiative.com/advisories/ZDI-21-1115/">https://www.zerodayinitiative.com/advisories/ZDI-21-1115/</a>
CVE Allocate Date	CVE	2021-07-14
NVD Publish Date	NVD	2021-09-29
NVD Last Modified	NVD	2021-10-02
NVD Vulnerable Configuration	NVD CPE	cpe:2.3:a:trendmicro:serverprotect:5.8:.*:.*:.*:emc:.*: cpe:2.3:a:trendmicro:serverprotect:5.8:.*:.*:.*:netapp:.*: cpe:2.3:a:trendmicro:serverprotect:5.8:.*:.*:.*:netware:.*: cpe:2.3:a:trendmicro:serverprotect:5.8:.*:.*:.*:windows:.*: cpe:2.3:a:trendmicro:serverprotect:6.0:.*:.*:.*:storage:.*:
NVD CWE ID	NVD	<a href="#">CWE-287</a> .
NVD CVSS V3 Score	NVD	Base score is 9.8.
NVD CVSS V3 Vector	NVD	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H.
NVD CVSS V2 Score	NVD	Base score is 10.
NVD CVSS V2 Vector	NVD	CVSS:2/AV:N/AC:L/Au:N/C:C/I:C/A:C.

- *CVSS V2 Exploitability Property*  $P_{Exploit}^{v^2}$  quantifies the likelihood as well as the effort and intricacy to be invested for exploiting a component that would be exposed to a given vulnerability. This exploitability property utilizes a set of three metrics for measurement, namely *AccessVector* (*AV*), *AccessComplexity* (*AC*) and *Authentication* (*Au*). The Access Vector metric measures the likelihood for an attack scenario targeting the component to occur through this vulnerability, through a physical-based, an adjacent network based, or a network-based access. The effort that needs to be invested may vary across these scenarios, quantified as part of the Access Complexity metric. The Authentication metric evaluates the level of authentication an attacker needs to require before launching an attack.
- *CVSS V2 Impact Property*  $P_{Impact}^{v^2}$  groups metrics along the (CIA) triad, to quantify the magnitude of potential losses of *Confidentiality* (*C*), *Integrity* (*I*) and/or *Availability* (*A*). Measurements along these metrics categorize the severity levels impacted by the vulnerability as none- (*N*), low- (*L*) or high- (*H*).

Next, we explain the base score properties of CVSS Version 3 or V3. These base properties are further grouped under three classifications, namely exploitability  $P_{Exploit}^{v^3}$ , scope  $P_{Scope}^{v^3}$ , and impact  $P_{Impact}^{v^3}$  properties. Impact property of CVSS V3 share the same metrics as the impact property of CVSS V2, and therefore would not be discussed.

- *CVSS V3 Exploitability Property*  $P_{Exploit}^{v^3}$  combines the following metrics: *AttackVector* (*AV*), *AttackComplexity* (*AC*), *PrivilegesRequired* (*PR*) and *UserInteraction* (*UI*). The Attack Vector and the Attack Complexity metrics measure similar aspects as the Access Vector and Access Complexity metrics of CVSS V2. Along the path of an attack scenario, some credentials or privileges may be required. The level of these requirements is measured by the Privileges Required metric, for an agent with authority to be granted access to the component. And, the level of participation that is expected in order to exploit and compromised the vulnerable component is measured by the User Interaction metric.

- *CVSS V3 Scope Property*  $P_{Scope}^{v3}$  measures the propagation of a vulnerability from a targeted component to eventually grant access to others within an asset configuration, through a metric named *ScopeChange* (or S). The Scope metric is used to measure the extent to which other components than the vulnerable one, can be accessed.
- *CVSS V3 Impact Property*  $P_{Impact}^{v3}$  adopts the same metrics as the CVSS V2 impact property.

Temporal properties refer to the dynamic characteristics of a vulnerability, which changes over time. This is because the exploits complexity of a vulnerability might change due to exploit techniques used by attackers as well as patch-techniques' enhancement from security experts. The exploit-techniques' evolution can be measured by *ExploitCode-Maturity*, while the patch-techniques' enhancement can be measured by *Remediation-Level*. Another interesting character of a temporal property is its vendor dependence, which affects the degree of credibility of the vulnerability instance using the metric *ReportConfidence* (Wang, Zhang, and Xia, 2008).

Environment properties emerge from the dynamic characteristics of a vulnerability that vary across different deployment environments. This is because the vulnerable asset may be deployed in a different location of the infrastructure that influences the *centrality* and functional *importance* aspects within the CPS infrastructure, which reflects the magnitude of induced consequences from successfully exploiting the vulnerability. The environmental property can be measured by *SecurityRequirements*, characterized by *ModifiedBaseMetrics* that factor in weights towards base metrics.

CVSS combines the above properties to infer vulnerability level rating its severity based on a rule-based algorithm, which is further depicted in Expression 2.1 that use measurements of *Exploitability*, *Scope* and *Impact* property metrics to generate the *Base* score of a vulnerability. Considering a component  $c$  of a CPS asset  $C$ , exploitability and impact property measurements are extracted as illustrated Expressions 2.2 and 2.3, separately. To infer a score of a vulnerability  $v$  for a component  $c$ , a function measures the corresponding base properties:  $f_{Exploit}$ ,  $f_{Scope}$  and  $f_{Impact}$ , as illustrated by the  $f_{Base}$  function illustrated by Expression 2.1.

$$f_{Base}^{v2} = (P_{Exploit}^{v2}, P_{Impact}^{v2}), \text{ or } f_{Base}^{v3} = (P_{Exploit}^{v3}, P_{Scope}^{v3}, P_{Impact}^{v3}) \quad (2.1)$$

In Expression 2.1, vulnerability measurements vector  $v_i$  are collected for component  $c_i$  by Expressions 2.2 and 2.3.  $f: A \rightarrow B$  means a functional association.

$$f_{Exploit}: C \rightarrow P_{Exploit} \quad (2.2)$$

$$f_{Impact}: C \rightarrow P_{Impact} \quad (2.3)$$

Taking the vulnerability instance *CVE-2021-37172* (2021) as an example, this vulnerability instance affects *Siemens* PLC product running SIMATIC S7-1200 CPU family with firmware version number 4.5.0 (i.e., the vulnerable component), by allowing a threat agent to bypass authentication and download arbitrary programs to this PLC (i.e., the vulnerable CPS asset). This vulnerability has a CVSS version 3 base score of 7.5, which is further composed of an exploitability score of 3.9 as well as an impact score of 3.6.

CVSS maps a numerical score to a qualitative severity level. Vulnerability severities are rated as '*low*' (V2 score  $\in [0.0 - 3.9]$ ), '*medium*' (V2 score  $\in [4.0 - 6.9]$ ), or '*high*' (V2 score  $\in [7.0 - 10.0]$ ) according to the *CVSS V2 Documentation* (2022). While according to the *CVSS V3 Documentation* (2022), vulnerability severities are rated as '*none*' (V3 score = [0.0]), '*low*' (V3 score  $\in [0.1 - 3.9]$ ), '*medium*' (V3 score  $\in [4.0 - 6.9]$ ), or '*high*' (V3 score  $\in [7.0 - 8.9]$ ), or '*critical*' (V3 score  $\in [9.0 - 10.0]$ ).

## 2.2.2 ENUMERATIONS

Open enumerations are advocated to itemize system weakness, instances of malicious attacks' categorizations, and vulnerable system configuration to guide a cybersecurity-analysis process. This thesis presents more details of five enumerations briefly introduced in section 1.2.

This sub-section introduces several major cybersecurity enumerations. These standards include respectively a taxonomy from *cwe*, threat categorization from the website *CVE Details* (2022) and attack patterns from *Common Attack Pattern Enumeration and Classification (CAPEC)* (2022), as well as *Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK)* (2022). Besides, security analysts need to match a current system configuration with retrieved vulnerability reports from enumerations like vulnerable product dictionary from *CPE* and recommended secure system configurations from *Common Configuration Enumeration (CCE)* (2022).

### (i) Common Weakness Enumeration (*CWE*) and Weakness Labels

*CWE* provides a comprehensive dictionary and classification taxonomy of known vulnerabilities. *CWE* version 4.6 currently provides three concept-views on the *CWE* official site, i.e., views by *Software Development*, by *Hardware Design*, and by *Research Concepts*. In total, there are 924 documented weaknesses in the *CWE* database (last checked November 3, 2021). These weaknesses have been used to construct 44 different views or ontologies, some of which cross-reference each other. Different ontologies are introduced either as rigorous academic studies or ad hoc collections based on experience. One example is the direct mapping from *CVE-ID(s)* to *CWE-ID(s)* that is carried out manually by security analysts like *NVD* analysts. Another well-known attempt of *CWE* ontology is the installment of the *Open Web Application Security Project Top Ten 2021* (2022) list that groups and categories *CWEs* and selects the top ten weaknesses with the highest exploitability, impact and incident rate in web application security. Platforms like *Computer Aided Integration of Requirements and Information Security* (2022) and *IriusRisk* (2022) employ *CWE* for vulnerability patterns analysis. CAIRIS requires manual definitions for each vulnerability and attack while selecting templates imported from *CWE*. Similarly, *CWE* libraries are loaded at application startup time and are accessible in *IriusRisk*, to provide references to threat modeling.

The *CWE* hierarchy follows a top-down format from *Pillar* (which has a set of 10 *CWE-IDs*), to *Class* (which has a set of 100 *CWE-IDs*), then to *Base* (which has a set of 533 *CWE-IDs*), and finally to *Variant* (which has a set of 297 *CWE-IDs*). *Class*-level *CWE* item provides a broad overview of a vulnerability type and is independent of any implementation. *Base-* and *Variant*-level *CWE* items provide finer granularity vulnerability types. *CWE* adopts a hierarchical structure where the top categories contain tree-structure patterns. Non-root lower level *CWE* nodes inherit the characteristics of the parent nodes (Aghaei, Shadid, and Al-Shaer, 2020). Although tree-based ontologies are useful for understanding and clustering weaknesses (Han et al., 2018), the applications of ontology-based frameworks are limited due to the complexity of extracting and cata-

logging *CWEs* from vulnerability repositories (Ruohonen and Leppänen, 2018).

The current *CWE* version 4.6 (last checked November 3, 2021) contains a list of 1250 *CWE* items that are identifiable with *CWE-ID*s. Each *CWE* item further contains names, descriptions, views, demonstrative examples, potential mitigations, memberships, and may have mappings to *CAPEC-ID*s and references. In total, there are 323 *CWE* items that have related attack patterns with *CAPEC-ID*s. The remaining 927 *CWE* items have no correlated attack patterns. For instance, *CWE-369*, *CWE-476* and *CWE-787* (shown later in Table 2.2) have no related attack type.

*CWE* entries address key features of exploitability and consequences of a vulnerability. Recall the vulnerability instance *CVE-2021-36745* (shown in Table 2.2), its assigned *CWE-ID* is '287' with a name as '*Improper Authentication*'. The supportive information of *CWE-287* undertakes additional features that are not included in the original report of this instance, and include: (i) What is the weakness? (The software does not prove or insufficiently proves that an actor's claim is correct); (ii) How likely is this weakness exploited? (High); (iii) What are the common consequences? (Read application data, or gain privileges, or assume identity, or execute unauthorized commands).

#### (ii) Common Attack Pattern Enumeration and Classification (*CAPEC*)

*CAPEC* sets forth a common set of identification for cyber attack patterns. *CAPEC* version 3.6 currently provides two views on the *CAPEC* site (last checked November 3, 2021), i.e., Mechanisms of Attack and Domains of Attack. *CAPEC* adopt similar hierarchical patterns as *CWE*. The *CAPEC* hierarchy follows this top-down format from View to Category, to Meta, then to Standard, and finally to Detailed. There exists the potential for some attack patterns to align with more than one category depending on one's perspective. *Techniques* refer to the actions that adversaries need to perform in order to accomplish goals, which are translated into various tactics. The current *CAPEC* version 3.6 contains a list of 546 attack patterns, each of which has a *CAPEC-ID*, name and description for each ID, the likelihood of an attack, typical severity, dependent attack pattern(s), views, prerequisites, mappings to *CWE-ID*(s) and all required external references. Currently, there is no explicit link between *CVE-ID*(s) and *CAPEC-ID*(s), which is in fact correlated through *CVE->CWE* and *CWE->CAPEC*. For instance, *CWE-287* discussed in the previous section is correlated to ten different *CAPEC-ID*s such as *CAPEC-114* ("Authentication Abuse") and *CAPEC-151* ("Identity Spoofing"). Additionally, some *CAPEC* entries are further correlated to the *ATT&CK* taxonomy, of which the details are seen in *CAPEC View With Entry ID 658* (2022).

#### (iii) Adversary Tactics, Techniques and Common Knowledge (*ATT&CK*)

*ATT&CK* is adopted to build threat awareness into cybersecurity solutions. *ATT&CK* contains a comprehensive matrix of known adversary tactics and techniques used during cyberattacks (Strom et al., 2018) (Al-Shaer, Spring, and Christou, 2020). It is interpreted as tactics that describes the tactical goal or the reason for an adversary to perform an action, techniques that suggest how an adversary performs a malicious action, and procedures that hint a specific implementation an adversary uses for techniques. There are also sub-techniques that are specific categorical malicious behaviors. *ATT&CK* provides a structured taxonomy to describe adversary behaviors in three major domains, namely *Enterprise*, *Mobile* and *ICS* domains. The *Enterprise* domain covers adversary tactics and techniques on standard IT systems like Linux or Windows based OS. The *Mobile* domain contains attacks on mobile devices. And lastly, the *ICS* domain includes attacks targeting CPS based systems, particularly ICSs. The latest version of *ATT&CK* is version 10 (last checked November 3, 2021). Unlike *CWE* and *CAPEC*, *ATT&CK* doc-

uments do not cover *CVE* examples for each *ATT&CK-ID*. Nevertheless, mappings from *CVE-ID(s)* to *ATT&CK-ID(s)* are partially accessible through the pipeline of *CVE->CWE*, *CWE->CAPEC*, and finally *CAPEC->ATT&CK-ID(s)*. One example is the mapping of *CVE-2012-0931* to *ATT&CK-T1548*, as shown in Table 2.3.

Table 2.3: Example of *CVE-ID* mapping to *ATT&CK*

<i>CVE-ID</i>	<i>CWE-ID(s)</i>	<i>CAPEC-ID(s)</i>	<i>ATT&amp;CK-ID(s)</i>
<i>CVE-2021-3440</i>	<i>CWE-269</i> (i.e., Improper Privilege Management)	<i>CAPEC-122</i> (i.e., Privilege Abuse)	N/A
		<i>CAPEC-233</i> (i.e., Privilege Escalation)	<i>T1548</i> (i.e., Abuse Elevation Control Mechanism)
		<i>CAPEC-58</i> (i.e., Restful Privilege Elevation)	N/A

#### (iv) *cvedetails.com* and Threat Labels

The website *cvedetails.com* provides 13 threat types that a vulnerability instance may be exposed to, namely *DoS*, *Code Execution*, *Overflow*, *Memory Corruption*, *SQL Injection*, *XSS* (or cross-site scripting), *Directory Traversal*, *HTTP Response Splitting*, *Bypass Something*, *Gain Information*, *Gain Privileges*, *cross-site request forgery* (CSRF), and *File Inclusion*. Note that these 13 threat types are regarded as vulnerability types in the *cvedetails.com* website. This thesis defines these types as threat types as they are closer to the symptoms than to the root causes. Figure 2.6 shows the distributions of threat types. The top 3 threat types in terms of the number of vulnerability instances are *Code Execution*, *DoS* and *Overflow* with more than 39k, 26k and 20k instances, separately (last checked November 3, 2021).

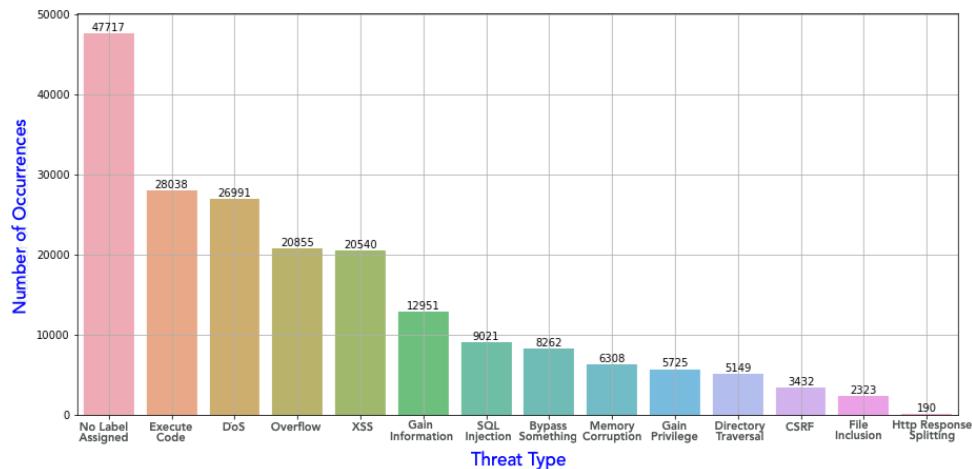


Figure 2.6: Threat type (assigned by *cvedetails.com*) distributions for vulnerabilities published from 1999 to November 3, 2021

#### (v) Common Platform Enumeration (*CPE*)

*CPE* provides a vulnerable product dictionary. The current version of *CPE* is 2.3, which is included in the Security Content Automation Protocol (SCAP) standard version 1.2 by the U.S. NIST for cybersecurity automation. The current *CPE 2.3* naming specification

follows a formatted string binding and an abstract logical construction known as well-formed *CPE* name. One example is VMWare vCenter Server 6.7 that is expressed as *cpe:2.3:a:vmware:vcenter\_server:6.7:-:.\*:.\*:.\*:.\**. This string binding is interpreted as *wfn:[part="a", vendor="vmware", product="vcenter\_server", version="6.7"]*. Here *part* can have three values, namely "*h*" that refers to hardware device, "*o*" that refers to operating system, or "*a*" that refers to software application. Note that the *CPE* metadata also indicates the applicable vulnerable system configurations. For example, vulnerability instance *CVE-2020-0964* (2020) describes a remote code execution vulnerability in Windows Graphics Device Interface, which applies to a list of servers like Windows Server 2016 and Windows Server 2019. Another vulnerability instance *CVE-2020-0966* (2021) indicates a remote code execution vulnerability in the VBScript, which is only applicable to Windows Server 2016 and 2019 when Internet Explorer 11 is running on the server. Note that vendor names in *CPE* metadata may appear with variations. For example, the vendor *Schneider Electric SE* has variant forms like "schneider-electric", "chneider-electric", and "schneider-electric".

### 2.2.3 VULNERABILITY REPOSITORY

Security-related data can usually be obtained in two ways: (i) direct access using software like vulnerability scanner; and (ii) indirect access using existing online public datasets. Direct data collection approaches are suitable for short-term analytics, or a comparatively small amount of collected data (Xin et al., 2018). There exist other indirect offline accesses to security datasets. However, those data accesses usually require stakeholders' authorization or ethical concerns. This thesis concentrates on online public datasets.

This sub-section mainly introduces seven enumerations. *CVE*, *NVD*, and *CERT Coordination Center (CC) Vulnerability Note Database (VND)* (2022) have been recognized as standard resources for security analysis that publish vulnerability reports regularly. Additional available sources of security-related data could be gathered in online forums such as *Exploit Database* (2022) and *SecurityFocus Forum* (2021), third-party analysts such as *ICS-CERT*, *Shodan Database* (2022) (or sentient hyper-optimized data access network) and *VulDB*, as well as manufacturer websites such as *MSRC*. The *SecurityFocus* archive is shut down in January 2021 and not accessible. *SecurityFocus* is included in this chapter as some of the experiments conducted earlier involve the *SecurityFocus* dataset.

#### (i) Common Vulnerability Enumeration (*CVE*)

*MITRE* Corporation publishes *CVE* and assigns an identifier to each discovered vulnerability. In addition, it maintains a publicly accessible database of all identifiers through *CVE Numbering Authorities* (2022). A typical *CVE* entry includes the following fields: a unique identifier, a brief description of the reported vulnerability, and any pertinent references about the vulnerability, as illustrated in the example of vulnerability instance *CVE-2021-37172* (2021) in Figure 2.7. The unique *CVE* identifier, or *CVE ID*, is the key that differentiates one security vulnerability from another. In doing so, *CVE IDs* provide a reliable way of communicating across these different databases to get more information about the reported security flaws. *CVE* unstructured data can be downloaded in CSV, JSON (provided by the *CVE Automation Working Group* (2022)), XML, or a XML-based *Common Vulnerability Reporting Framework* (2021) (CVRF) formats.

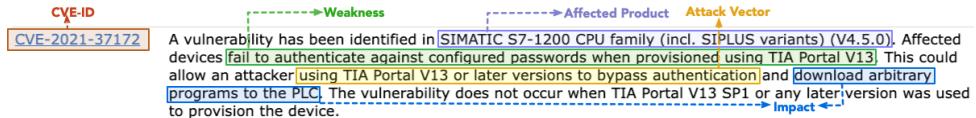


Figure 2.7: *CVE-2021-37172* as an example of published vulnerabilities in *CVE*

#### (ii) *CERT/CC Vulnerability Note Database (VND)*

Carnegie Mellon University initialized and maintained *CERT/CC VND* to provide information about software vulnerabilities, with more than 3 500 vulnerability notes till October 2021 (Madnick, Li, and Choucri, 2009). The *CERT Coordination Center Vulnerability Data Archive* (2020) is incomplete, as addressed by the organizer that only around 6% of received reports have been analyzed, published and disclosed as *CERT Vulnerability Notes* (2020). *CERT/CC* also develops a Python-based web platform named *VINCE* for easier data retrieval and interactions. *VINCE API* delivers both raw JSON data and vulnerability cases in CVRF format. Each vulnerability note in *CERT/CC* contains an ID (e.g., *VU#883754*), an overview, a detailed description, the impact of this vulnerability, vendor statements (not mandatory), as well as mitigations and references. *CERT/CC* vulnerability archive is independent from *CVE*, even though some vulnerability notes in *CERT/CC* provide cross-references to *CVE* vulnerabilities, i.e., *CVE* IDs.

#### (iii) *ExploitDB*

*Exploit Database* (2022) discloses public exploits, PoC and shellcode to demonstrate how a vulnerability can be exploited (Almukaynizi et al., 2017). An exploit normally contains a title, description, PoC code, and metadata like exploit ID, *CVE-ID*, and *CWE-ID*. According to a study by Sun et al. (2021), 73.5% exploits were announced publicly in *ExploitDB* at least one day earlier than the published date of the corresponding *CVEs*. In contrast, around 28% exploits disclosed in *ExploitDB* have no assigned *CVEs*. Figure 2.8 presents an example of published exploit in *ExploitDB* without any correlated *CVE-ID*. This exploit report also clarifies the affected product, version, potential impact and PoC exploit.

An interesting investigation from Cyentia Institute and Security (2018) shows that the rate of active exploitation is correlated to the date when the vulnerability is published as well as the date when the relevant exploit is published. They further argue that “*roughly two out of every three exploited CVEs have associated published code. When an exploit code is published, the chance to observe an exploitation vector in the wild is seven times higher than without a published exploit code*”.

#### (iv) *SecurityFocus*

*SecurityFocus Forum* (2021) is a widely used vulnerability database and also features a security news portal (Fang et al., 2020). Besides vulnerability descriptions, *SecurityFocus* also addresses whether a vulnerability has a PoC exploit. Note that *SecurityFocus* is not dependent upon *CVE* data sources (Rodriguez et al., 2018). Actually, a BugTraq vulnerability report may refer to several *CVE* vulnerability instances. A statistic analysis by Fang et al. (2020) highlights that although the amount of vulnerabilities reported in *SecurityFocus* is less than the number of vulnerabilities found in *NVD*, the fraction of exploited vulnerabilities in *SecurityFocus* (37.008%) is much higher than the proportion in *NVD* (6.676%). Fang et al. (2020) also observe that the vulnerability reports in *SecurityFocus* contain higher coverage and more reference significance in predictive cy-

SAP B2B / B2C CRM 2.x < 4.x - Local File Inclusion Weakness Type

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
44655	N/A	RICHARD ALVIAREZ	WEBAPPS	LINUX	2018-05-18
EDB Verified: ✅		Exploit: ⚡ / {}		Vulnerable App:	
<pre># Title: SAP B2B / B2C CRM 2.x &lt; 4.x - Local File Inclusion # Application [SAP B2B OR B2C is CRM] # Versions Affected: [SAP B2B OR B2C is CRM 2.x 3.x and 4.x with # Backend R/3 (to icss_b2b)] <b>Affected Product</b> # Vendor URL: http://SAP.com # Bugs: [SAP LFI in B2B OR B2C CRM] # Sent: 2018-05-03 # Reported: 2018-05-03 # Date of Public Advisory: 2018-02-09 # Reference: SAP Security Note 187025566 # Author: Richard Alviarez  # 1. VULNERABLE PACKAGES # [SAP LFI in B2B OR B2C CRM v2.x to 4.x] <b>Affected Product</b> # Other versions are probably affected too, but they were not checked.  # 2. TECHNICAL DESCRIPTION # A possible attacker can take advantage of this vulnerability # to obtain confidential information of the platform, # as well as the possibility of writing in the logs of the # registry in order to get remote execution of commands and # take control of the system.</pre> <p><b>Impact</b></p>					
<p># 3. Steps to exploit this vulnerability</p> <p>A. Open <a href="https://SAP/{name}_b2b/initProductCatalog.do?forwardPath=/WEB-INF/web.xml">https://SAP/{name}_b2b/initProductCatalog.do?forwardPath=/WEB-INF/web.xml</a></p> <p>Other vulnerable parameters:</p> <pre>https://SAP/{name}_b2b/CatalogClean.do?forwardPath=/WEB-INF/web.xml https://SAP/{name}_b2b/IbaseSearchClean.do?forwardPath=/WEB-INF/web.xml https://SAP/{name}_b2b/ForwardDynamic.do?forwardPath=/WEB-INF/web.xml page on SAP server</pre> <p>B. Change parameter {name} for example icss_b2b or other name....</p> <p>C. Change "/WEB-INF/web.xml" for other files or archives internal.</p> <p><b>Exploit</b></p>					
<pre># 4. Collaborators # - CuriositySec # - aDoN90 # - Vis0r</pre>					

Figure 2.8: EDB-ID 44655 as an example of published exploits in ExploitDB

bersecurity analysis, leading to their experiment results where *SecurityFocus* performs better than *NVD* in an actual environment.

#### (v) US-CERT

The United States *Computer Emergency Response Team* (2022) provides two major advisories towards CI and key resources cyber information sharing. One is *ICS-CERT Advisories* (2022) advisory that focuses on control systems' security. The other is *CERT Alerts* (2022) that provides an early warning of a single specific threat or vulnerability expected to have a significant impact once exploited. The *ICS-CERT* advisories add further analysis on reported vulnerabilities in CVE, particularly on risk evaluation, affected products, and mitigations such as workarounds or official patches.

#### (vi) Shodan

*Shodan Database* (2022) is a search engine that scans the internet to find open ports. It is also a data source mainly targeting CPS or IoT security, including SCADA (Bodenheim et al., 2014). CPS and IoT systems include devices like webcams, routers, and servers. Relevant information like ports and vulnerabilities of these devices can be fetched through *Shodan* website or *Shodan API*. Interestingly, these are currently internet-connected devices, sending (public) live data from different locations across the world. Unlike *NVD*, where vulnerability reports are published, *Shodan* crawls IP addresses, made available on device respective websites and APIs. Returned data from *Shodan* can be cross-referenced with *NVD* for vulnerability analysis (Fagroud et al., 2020). To get the API key of *Shodan*, the user needs to hold either an enterprise account or an academic account.

### (vii) National Vulnerability Database (NVD)

*National Vulnerability Database (NVD)* (2022) is built upon *CVE* entries to provide enhanced information for each entry, such as severity scores (calculated based on *CVSS* standard) and impact ratings. Newly disclosed *CVE* entries typically appear in *NVD* within an hour, as written in *CVEs and the NVD Process* (2022). *NVD* also provides advanced searching features such as by OS, by vendor name, by product name, by version number, and by vulnerability type and severity (Spanos, Angelis, and Toloudis, 2017). The vulnerability severity score is calculated following the *CVSS* version 3 and version 2 standards. *NVD* converts the unstructured *CVE* data into structured JSON or XML formats (Anwar et al., 2020). *NVD* adopted a subset of *CWE* to categorize vulnerabilities. In 2019 September, *NVD* upgraded the *CWE* subset mapping to the *CWE-1003* list. Also, since 2019 November, *NVD* consumed CNA's *CVSS* scores and *CWE* classes, together with *CVSS* scores and *CWE* data assigned by *NVD* (Byers, Waltermire, Turner, et al., 2020).

(a) Screenshot of Vulnerability CVE-2021-1529 on Oct 22, 2021

**CVE-2021-1529 Detail**

**UNDERGOING ANALYSIS**

This vulnerability is currently undergoing analysis and not all information is available. Please check back soon to view the completed vulnerability summary.

**Description**

A vulnerability in the CLI of Cisco IOS XE SD-WAN Software could allow an authenticated, local attacker to execute arbitrary commands with root privileges. The vulnerability is due to insufficient input validation by the system CLI. An attacker could exploit this vulnerability by authenticating to an affected device and submitting crafted input to the system CLI. A successful exploit could allow the attacker to execute commands on the underlying operating system with root privileges.

**Severity** CVSS Version 3.x CVSS Version 2.0

**CVSS 3.x Severity and Metrics:**

- NIST:** NVD
- Base Score:** N/A
- NVD score not yet provided.

- CNA:** Cisco Systems, Inc.
- Base Score:** T.8 HIGH
- Vector: CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

**QUICK INFO**

CVE Dictionary Entry: CVE-2021-1529  
NVD Published Date: 10/20/2021  
NVD Last Modified: 10/21/2021  
Source: Cisco Systems, Inc.

(b) Screenshot of Vulnerability CVE-2021-1529 on Oct 27, 2021

**CVE-2021-1529 Detail**

**Current Description Affected Product**

A vulnerability in the **CLI** of Cisco IOS XE SD-WAN Software could allow an **authenticated, local attacker** to **execute arbitrary commands with root privileges**. The vulnerability is due to **insufficient input validation by the system CLI**. An attacker could exploit this vulnerability by authenticating to an affected device and submitting crafted input to the system **CLI**. A successful exploit could allow the attacker to **execute commands on the underlying operating system with root privileges**.

**Impact**

**Attack Vector/type**

**View Analysis Description**

**Severity** CVSS Version 3.x CVSS Version 2.0

**CVSS 3.x Severity and Metrics:**

- NIST:** NVD
- Base Score:** T.8 HIGH
- Vector: CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

- CNA:** Cisco Systems, Inc.
- Base Score:** T.8 HIGH
- Vector: CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

**CVSS Score Assigner**

**CVSS V3 Base Score**

**QUICK INFO**

CVE Dictionary Entry: CVE-2021-1529 → **CVE-ID**  
NVD Published Date: 10/20/2021  
NVD Last Modified: 10/26/2021 → **Document Tracking**  
Source: Cisco Systems, Inc.

**References to Advisories, Solutions, and Tools**

Hyperlink	Resource	Vendor Advisory	References from Vendor/Analyser
<a href="https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-sd-wan-rhpBE34A">https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-sd-wan-rhpBE34A</a>			

**Weakness Enumeration**

CWE-ID	CWE Name	Source	CWE Weakness Type
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	NIST Cisco Systems, Inc.	

Figure 2.9: *CVE-2021-1529* as an example of published vulnerabilities in *NVD*

The vulnerability analysis process takes time. One example is the vulnerability instance *CVE-2021-1529* disclosed in *CVE* on October 20th, 2021. This vulnerability is assigned a *CVSS* version 3 base-score of 7.8 by the vendor *CISCO*, while waiting for a *CVSS* score

to be assigned by NVD analysts manually, as shown in Figure 2.9 (a). NVD then assigns a score with the same value as CISCO-assigned score to this vulnerability on October 26, 2021, as shown in Figure 2.9 (b). In addition, the vulnerability-related features, such as affected product, CWE-ID and references, are also marked in Figure 2.9.

## 2.3 VULNERABILITY LIFECYCLE

Figure 2.10 presents a typical vulnerability lifecycle, including Creation, Discovery, Exploit, Disclosure, Countermeasure Available and Patch stages (Joh and Malaiya, 2011). An application software with bug or backdoor may give rise to design and implementation weaknesses. When such weaknesses are discovered, attacks may be triggered to exploit the vulnerabilities. A thorough understanding of the vulnerability lifecycle enhances the ability to defend against potential threats. *Discovery stage* refers to the first time when a vulnerability is discovered and reported (usually in a private group). *Exploit stage* is the time when an exploit for a vulnerability occurs. *Disclosure stage* is defined as the time when a vulnerability is publicly disclosed by a certain party. PoC or working code for exploiting a vulnerability may be available before or after the disclosure. *Countermeasure Available stage* refers to the time when a fix or a workaround by a third party is available, which is contrasted to *Patch stage* when the vendor or the originator releases an official fix. Between discovery and disclosure stages, the vulnerability instance is only known to private groups. While after disclosure, the knowledge is known by the public. Therefore, the time span between Discovery and Disclosure increases prospects for exploits to occur, while post-disclosure leads to patching opportunities (Arora et al., 2006).

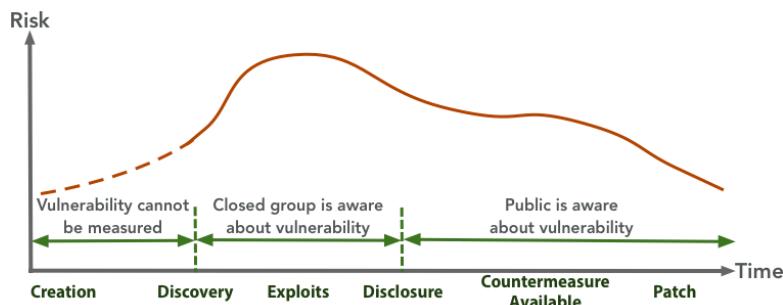


Figure 2.10: Vulnerability lifecycle adapted from Frei et al. (2006)

Early detection or identification of vulnerabilities minimizes the *window of exposure* (Frei et al., 2006), which enhances the security of a system. The tool and method used for vulnerability identification vary according to the application context, as demonstrated by Antunes and Vieira (2014). Normally, two types of approaches are advocated to pinpoint vulnerabilities and unravel their corresponding attributes, namely static (passive) vulnerability identification and live (run-time) vulnerability detection (Neuhaus and Zimmermann, 2010) (Austin, Holmgreen, and Williams, 2013). Static approaches include penetration testing (Bertoglio and Zorzo, 2017), vulnerability code pattern recognition, or passive reconnaissance, especially in software debugging or web-based environment (O'Hare, Macfarlane, and Lo, 2019). Software vulnerability differs to other vulnerability categories (e.g., hardware vulnerability) in that software vulnerability is closer to *fault*, *flaw* or *bug* (Ghaffarian and Shahriari, 2017). Antunes and Vieira (2014) conduct an

evaluation of eight different web-services vulnerability-detection tools, including four penetration testing tools, three static code analysis tools and one anomaly detection tool. The run-time vulnerability detection approach involves either a (nearly) real-time system-wide vulnerability scanner or anomaly detection system (Farris et al., 2018). Vulnerability scanners have been adopted to capture attackers' activities in mainly network based environments. For example, Holm et al. (2011) collect information by a network-scanner. Then, collected information is synchronously compared with a database of vulnerability-signatures to identify security patterns. Static- and live-vulnerability identification methods are combined to produce multiple sources of correlated data in some studies like (Jajodia et al., 2011).

Vulnerability-categorization approach requires the exploration of reported vulnerabilities in a vulnerability database to identify vulnerability patterns, such as the methods proposed by Huang et al. (2010), Almukaynizi et al. (2017), Bozorgi et al. (2010) and Christey and Martin (2007). As discussed in the earlier section, databases like *CVE* accumulates vulnerability reports for more than 23 years (from 1999 till now), and could provide a basis for offline security-practices or vulnerability-trends analysis (Chang et al., 2011). *CWE* provides a comprehensive dictionary and classification taxonomy of known vulnerabilities. It adopts a top-down hierarchical structure where the top categories can contain tree-structure patterns. Although tree-based ontologies are useful for understanding and clustering weaknesses (Han et al., 2018), the applications of ontology-based frameworks are limited due to the complexity of extracting and cataloging *CWEs* from vulnerability repositories (Ruohonen and Leppänen, 2018). The documented weaknesses in *CWE* have been used to construct different ontologies in which references may cross each other. Some other efforts into vulnerability classification include manual categorization of operation-system vulnerabilities by Abbott et al. (1976), as well as more advanced classification methods based on ML algorithms like the works by Na, Kim, and Kim (2016), Chen et al. (2019) and Huang et al. (2019).

Vulnerability assessment plays a key role in patching prioritization and decision making (Hong, Kim, and Haqiq, 2014), which is categorized into qualitative methods and quantitative methods. Qualitative models primarily address relationships among vulnerability and risk and express vulnerability observations based on non-numerical data, such as those employed in risk management frameworks: CRAMM (Farquhar, 1991)(Yazar, 2002), OCTAVE (Alberts et al., 2003), CORAS (Fredriksen et al., 2002)(Den Braber et al., 2007), Aurum (Ekelhart, Fenz, and Neubauer, 2009), ISO/IEC 27005. Although qualitative approaches provide mitigation guidelines to manage risks through corresponding vulnerabilities (Shamala, Ahmad, and Yusoff, 2013), their abstract management procedures normally require experts' knowledge to manually regulate the assessment process (Ryan et al., 2012). Quantitative assessment, on the other hand, measures the likelihood and impact of a vulnerability to prioritize patching exercises (Allodi and Massacci, 2017). Some quantitative approaches such as the one from Houmb, Franqueira, and Engum (2010) employ the industrial standard *CVSS* as metrics to evaluate vulnerability scores. Another method for quantitative vulnerability assessment is to use a testbed for vulnerability and threat scenario simulation (Holm et al., 2015). For instance, Negi et al. (2019) develop a power-distribution testbed to evaluate some vulnerabilities in SCADA systems.

Vulnerability remediation may involve various strategies, such as updates, patches, and improving system access-control. Considering the expanding amount of vulnerabilities that need to be dealt with, a remediation strategy needs to focus on only a subset of vulnerabilities (Nayak et al., 2014). Such prioritization is usually connected with vulnerability severity and exploit likelihood (Fruhwirth and Mannisto, 2009) (Hong, Kim, and

Haqiq, 2014). Different approaches to metrics have been suggested to measure these two factors, including applying probabilistic metrics through threats, attacks and vulnerabilities tools (Jajodia et al., 2011), or directly using CVSS metrics (Houmb, Franqueira, and Engum, 2010) (Joh and Malaiya, 2011). Remediation focus should cover vulnerabilities that may result in security incidents and an efficiency attribute.

Several studies link remediation and patching to vulnerability lifecycle, especially the *window of exposure* between vulnerability discovery and available patch (Arbaugh, Fithen, and McHugh, 2000). Frei et al. (2006) suggest one of the earliest works on this topic. They collect vulnerability records from NVD, OSVDB (referring to the open source vulnerability database) and other online cybersecurity repositories like CERT and SecurityFocus, to empirically analyze the correlation between risk exposure with discovery-, exploitation-, disclosure- and patch-time.

Most existing vulnerability assessment techniques are based on scheduled analysis lifecycle, which can be weekly, monthly, or even yearly. This poses a limitation in security management of safety-critical systems such as CIs, as malicious attempts may use new exploits that occur between successive analysis milestones (Arbaugh, Fithen, and McHugh, 2000). Scheduled analysis may leave a gap between a vulnerability exploit occurrence and the deployment of an available patch (Shahzad, Shafiq, and Liu, 2012). On the contrary, a timely vulnerability-analysis lifecycle allows mitigations to occur within the time interval that span the discovery and disclosure of vulnerabilities. It also gives time for vulnerability patches to be available and deployed before the release time of exploits. Moreover, vulnerability assessment needs to adjust to fluctuating operational functionalities and changing threat-landscape, dynamically (Xu, 2019) (Xu, 2014). A closed-loop model that is sensitive to these dynamics is needed to provide near real-time risk-monitoring (Chen, Cho, and Xu, 2018). Such timely analysis method contrasts with traditional linear cycles within rigid management portfolios, which may tolerate security gaps and exploits occurrence beyond disclosure limits (Frei et al., 2006). The proposed approaches in this thesis short the *window of exposure* through timely vulnerability assessment.

# DATA DRIVEN VULNERABILITY ANALYSIS



# CHAPTER 3

## DATA DRIVEN VULNERABILITY ANALYSIS

This chapter elaborates on current state-of-the-art vulnerability analysis research. Namely, ontologies and data-driven cybersecurity approaches and their application to CIs protection are discussed. The goal is to highlight the benefits of data-driven vulnerability analytics for cybersecurity decision-making solutions.

### 3.1 VULNERABILITY DATA PREPARATION

Research efforts are rising in cyber intelligence exploiting increasing data from public vulnerability repositories and technical blogs (Liao et al., 2016). The increasing availability of public accessible cybersecurity databases provides a basis for further vulnerability analytics. Data is crucial for machine learning applications, as learning algorithms require adequate data to classify patterns or acquire predictive abilities. ML systems acquire knowledge by training, absorbing new data as they iteratively perfect their analysis. Consequently, the accessibility of precise and conflict-free data sources is crucial from a security standpoint.

#### 3.1.1 CORRELATIONS AMONG VULNERABILITY DATA

Databases such as the *CVE* repository accumulate vulnerability reports for around 20 years, which could provide a basis for vulnerability-trends analysis (Na, Kim, and Kim, 2016) (Russo et al., 2019). Tools like *CVE-Search* (2022) (referring to a web interface and API for CVE) and *Open Vulnerability and Assessment Language (OVAL)* (2022) are some of the common techniques used to identify and manage organization vulnerabilities in a vendor-independent environment. Application of these tools leads to streamlined approaches that retrieve vulnerabilities from public repositories in support of vulnerability assessment applications (Siboni et al., 2019). Besides, some commercialized vulnerability management and assessment products, such as *InsightVM* (2022) from Rapid7 Research, *Nessus* (2022) and *Tenable.io* from Tenable, deploy system scans that are matched with the CVE repository to monitor vulnerabilities and exposures of infrastructures. Although there have been more efforts by the industry to integrate security tools into a wider framework, this process is often very vendor-specific, leaving the burden on the security team for addressing ad-hoc security challenges.

Statistical interpretations of *CVE* and *NVD* datasets need to be combined with other live security-related data sources, such as Twitter, the dark web, and product vendors across deployed infrastructures, to raise reliability and precision of security indicators (Sauerwein et al., 2019). The reason is that using a single data source for cybersecurity analysis may introduce a bias. This is exemplified in the works of Chen et al. (2019) and Bullough et al. (2017), both of which extract vulnerability-related data by crawling Twitter and extracting tweets that contain *CVE* as a keyword. The goal of doing this is to connect related

analysis into broader statistical associations. For example, one study from Allodi and Massacci (2014) correlates NVD to data sources such as *Exploit Database* (2022), and *Symantec* (2022) *AttackSignature* and *ThreatExplorer*. Geer and Roytman (2013) correlate the NVD database with data sources such as ExploitDB and *Metasploit* (2022) to support penetration testers. However, further research that incorporates heterogeneous data from different databases while providing structured indicators is limited (Dong et al., 2019).

Various frameworks, ontologies and languages are proposed to enhance the interoperability of distributed cybersecurity data sources, with some examples such as SCAP, OVAL, CybOX, STIX, TAXII and CVRF (Brown, Gommers, and Serrano, 2015) (Kampanakis, 2014). Ontologies provide a formal and explicit way with structured indicators to specify concepts and relationships, and are therefore beneficial in cybersecurity information correlation. One representative of such works is the unified cybersecurity ontology proposed by Syed et al. (2016), which links multiple vulnerability databases (e.g., CVE, CWE, CAPEC), STIX (referring to structured threat information expression) version 1, as well as other ontologies. However, the reasoning ability of the unified cybersecurity ontology needs to be improved further. In a related research, Kotenko et al. (2018) propose an ontological storage that covers various cybersecurity databases by semantically interconnecting different cybersecurity objects, using ontology web language (OWL) and a description logic query language (based on SPARQL). The data sources implemented in their study are divided into groups to statistically analyze their connections, e.g., CAPEC, CVE and CWE in one group. However, the applicability and complexity of correlating all these databases are not considered. Doynikova, Fedorchenco, and Kotenko (2019) further extend the ontology with domain-specific security metrics, such as infrastructure metrics and event metrics. Still, information regarding the same vulnerability instance is not integrated into a single unified package from various data sources. Brazhuk (2019) uses similar techniques to correlate attack patterns in CAPEC and weakness concepts in CWE using OWL and SPARQL query languages.

Some works use ontological databases to support high-level reasoning, such as transitive closure and subsumption, to further improve threat intelligence. For example, Alqahtani, Eghan, and Rilling (2016a) use a symbolic-approach to improve knowledge management in cyber threat intelligence (CTI). Their framework applies a probabilistic semantic similarity measure to establish bi-directional traceability links between security databases, which is demonstrated using NVD and Maven build repository. Yet, pair-matching instances of these two data-sources and the following data-validation process are manually conducted, which might be prone to human errors. The same authors propose a unified ontology-based modeling approach for software vulnerability data sources in Alqahtani, Eghan, and Rilling (2016b). However, generating new knowledge through the manipulation of symbols is not included in this work. Nevertheless, reasoning capability is vital to support situation awareness in cybersecurity decision-making (Johnson et al., 2016a). Mavroeidis and Bromander (2017) summarize some of the common challenges in applying ontology for threat intelligence. They particularly highlight the lack of expressiveness and complication of combining different ontologies due to various terms associated with similar concepts. The authors also argue that most of the existing ontologies explicitly target specific threat/vulnerability categories, and therefore those ontologies limit the decision process to those threats/vulnerabilities only.

Nevertheless, there is a lack of common terminology that allows data interoperability and systematic analytics (Torkura, Cheng, and Meinel, 2016). Most of the existing studies are focused on data from a single database or have tailored their solutions to accommodate disparate data models and terminologies in use. Such methods usu-

ally require distinctive expertise for the employed data sources. Extensive remodeling or minor refinement is needed depending on the level of heterogeneity of these deployed sources. Translating standalone data objects and enumerations into a common data model (CDM) generates the possibility of study validation and systematic analytics across various data sources.

This thesis suggests a CDM to correlate diverse cybersecurity repositories or standards. This CDM allows us to collect unstructured and semi-structured data items from multiple public repositories, and to transfer them into structured data. Using this CDM, we integrate different data sources into semantic categories considering data instance co-occurrences. This approach contrasts with rigid traditional management portfolios, which may tolerate security gaps and exploits beyond disclosure limits. The strengths of our data curation approach are two-fold: (i) we collect unstructured and semi-structured data items from multiple public repositories, and transfer them into structured data; and (ii) we integrate these several sources into a single database using a CDM. In other words, vulnerability instances are collected synchronously from repositories, to assist operators in evaluating up-to-date vulnerability trends, and to further narrow the risk window induced by the discovered vulnerabilities.

### 3.1.2 VULNERABILITY DATA QUALITY

Recent developments in data analytics need data-driven vulnerability assessments, in which the data held and provided in the aforementioned repositories can be found in a variety of formats. These research and commercial solutions rely significantly on the integration multiple pieces of open-source vulnerability information, making it imperative to settle disagreements and prove the accuracy of such information.

However, some recent research works exemplify that the data quality of online vulnerability databases suffers from diversity, incompleteness, and redundancy, which hampers accurate vulnerability assessment (Jo et al., 2020) (Chaparro et al., 2017) (Anwar et al., 2020) (Dong et al., 2019). For instance, *SecurityFocus* and *ExploitDB* contain vulnerabilities that are yet to be reported in CVE. Thus, inferred decisions based on cybersecurity measurements need to include a wide range of vulnerability data repositories. Incomplete (Chaparro et al., 2017), outdated and incorrect vulnerability entries, may leave a risk window for potential zero-day attacks (Nappa et al., 2015) (Nappa et al., 2015). For instance, around 25% CVE reports that involve Google Chrome as affected products have incorrect Chrome version strings (Nguyen and Massacci, 2013). Additionally, varying CVE reporting templates make the reports too ambiguous to know what specific vulnerability it is referring to, forcing the reader to make assumptions about its meaning. For example, the attack types are reported using the consequence of the incident, the attack pattern, or the name of the device being targeted. CVE repository analytics involve recognizing vulnerability patterns based on severity scores. However, some CVE reports are neither scored, nor classified, which limits their mapping to some threat models and thus reduces mitigation effectiveness (Ladd, 2017). Recently reported instances are notably lacking such scoring valuations, yet they are crucial for computing severity scores that would otherwise result in poor patching decisions (Householder et al., 2017). Meanwhile, scanning through individual vulnerability reports may induce confusion since a single vulnerability report may address multiple vulnerability cases, e.g., a software embedded in multiple vendor components. Similarly, multiple reported cases may describe the same vulnerability, e.g., multiple reported components embedding a variation in the same software version. Therefore, vulnerability disclosure in these repositories cannot be directly used to identify known vulnerability instances in deployed system compo-

nents (Ruohonen, 2019). Instead, such instances need to be clustered into known patterns.

Nappa et al. (2015) make substantial efforts to identify and report discrepancies in NVD vulnerability entries. They crawl online security advisories in the *Mozilla* website and compared extracted vulnerable version ranges published in NVD. They summarize three main inaccurate issues of NVD entries, namely missing vulnerable versions, extraneous vulnerable versions, and missing vulnerable program file versions. The third issue is particularly true for Microsoft products such as *msword.exe*. Dong et al. (2019) conduct quantitative analysis on vulnerable software version inconsistencies between CVE and NVD. Their experimental data includes vulnerabilities reported from January 1999 to March 2018, which indicates that only 59.82% of the CVE summaries strictly match the standardized NVD entries. They also compare the matching rate of vulnerable software versions between NVD against CVE and five other public cybersecurity data sources, namely OpenWall, Security Focus, Security Focus Forum, Security Tracker, and ExploitDB. Their results show that the matching rate between NVD and ExploitDB is the highest, followed by the matching rate between NVD and CVE. Ruohonen and Leppänen (2018) apply information retrieval techniques as an alternative to support keyword-based searches in four data sources tracked in the Snyk vulnerability database. Based on their model performance using similarity metrics, they argue that neither CWE nor OWASP is ideal for constructing a reference corpus.

Jo et al. (2020) develop a tool, GapFinder, based on the text-mining technique to identify semantic inconsistencies and technical inconsistencies of open-source malware threat reports. By combining malware graph constructor with named entity recognition and relation extraction language processing techniques, their tool also addresses various language-specific issues and malware domain-specific issues. Anwar et al. (2020) perform a systematic evaluation of the consistency and completeness of NVD data feeds, based on which they further improve the quality of the data. Farhang et al. (2020) have investigated the degree to which Android smartphone vendors have included vulnerabilities in their security bulletins and found significant differences in the timeliness of reporting and the consistency of the reports.

This thesis includes three studies to analyze data quality issues in publicly accessible vulnerability repositories. Firstly, we carried out a baseline study to explore existing repositories and how to access them, as well as the data characteristics of one of the most commonly used repositories, *NVD*. Secondly, we facilitated a survey among CI stakeholders to understand the current usage of security databases and vulnerability scoring mechanisms in CI vulnerability analysis, and also to investigate different perceptions and user expectations of vulnerability repositories from CI cybersecurity practitioners. Thirdly, we performed a case study about the impact of data quality issues on vulnerability analysis of real-world IT and OT systems. These three studies look closely at whether or not current vulnerability repositories can help with the practice of CI vulnerability assessment. This further indicates the related challenges facing CI cybersecurity stakeholders.

## 3.2 MACHINE LEARNING FOR VULNERABILITY ANALYTIC

Emerging adversary strategies necessitate the development of new monitoring mechanisms to remediate the security gaps caused by the revealed vulnerabilities. Security intelligence appears to be an attractive methodology supported by vulnerability databases

and machine learning tools.

Contemporary advances in data-driven decision-making processes that are equipped with AI tools profoundly broaden the boundaries of traditional human decision-making processes (Buczak and Guven, 2015) (Larose, 2015), especially with ML techniques. Here, AI refers to the broad area of computer science that empowers machines with human intelligence (Smith and Eckroth, 2017). ML is a subset of AI that employs statistical learning algorithms to learn and improve from human-like experiences, and is usually categorized into supervised, unsupervised and reinforcement learning approaches (Xin et al., 2018). Even though ML for cybersecurity is a new field, it plays an important role in addressing certain problems related to cybersecurity (Chan et al., 2019) (Dilek, Çakir, and Aydin, 2015). For example, ML enhances automation and helps by reducing false positives in traditional cybersecurity solutions, and traditional labor-intensive data triage can be automated through event correlation rules to ease the burden of human analysts (Le, Chen, and Babar, 2021).

### 3.2.1 MACHINE LEARNING TECHNIQUES FOR VULNERABILITY ASSESSMENT

Kijewski and Pawliński (2014) suggest that automation is vital to coping with the flood of internal alerts and externally received information about vulnerabilities. ML techniques enhance automation and reasoning capabilities, improve performance, and support better prediction analysis of conventional cybersecurity solutions. To be effective, a ML approach requires big data whereby embedded data-mining algorithms empowered by ML techniques contribute to advanced analytics, that focus on extracting patterns (e.g., exploit patterns (Husari et al., 2017) and vulnerability patterns), or forecasting vulnerability trends (Buczak and Guven, 2015).

Retrieved information from vulnerability data sources supports further pattern recognition or trend analysis. The first large-scale vulnerability trend analysis uses the large and rapidly changing volume of *CVE* data as input, but manually selected *CWE* categories as topic-identifiers (Christey and Martin, 2007). However, this process could result in information loss, as not all vulnerabilities were accounted for in fixed data-sets. Therefore, it is desirable to automate the analysis process by translating natural language statements used in vulnerability reports into a machine-readable format and to apply text-mining techniques to extract patterned insights (Andrade et al., 2019). Traditional text-processing techniques fail to capture semantic similarities between words, and thus, the classification accuracy of resulting vulnerability patterns is decreased. Neuhaus and Zimmermann (2010) apply topic model based on latent dirichlet allocation (LDA) to analyze vulnerability trends using reports from *CVE*. A topic model can automatically extract topics from a textual corpus. Chen, Thomas, and Hassan (2016) further elaborate a survey of topic-modeling applications in software repositories.

Pattern-based methods are commonly used for vulnerability categorization and assessment (Andrade et al., 2019) (Torres, Comesáña, and García-Nieto, 2019). In these methods, feature-engineering approaches are used to extract textual descriptors that distinguish between vulnerabilities, detect anomalies, and generate a vectorial- or a graphical representation of vulnerability attributes used in pattern recognition processes. Extracting valid features is crucial to the effectiveness and applicability of ML. The choice of the features can be either determined manually by individual experts, or generated semi-automatically using natural language processing (NLP) techniques. A manual feature engineering process may benefit from experts' insights, while an automatic or semi-automatic process can effectively extract features from a larger corpus. Some of the

commonly-used NLP and text-mining techniques are summarized in Table 3.1 with related referenced papers. More details on related text-mining techniques are provided in Section 7.2.

ML techniques such as text-mining are applied to automatically classify disclosed vulnerabilities and guide predictive analytics of the security gap. Text-mining techniques have been widely used to collect cyber threat intelligence from cybersecurity repositories and technical blogs (Liao et al., 2016), to retrieve attack patterns (Husari et al., 2017), and to assess vulnerability severity (Khazaei, Ghasemzadeh, and Derhami, 2016) (Spanos, Angelis, and Toloudis, 2017). The effectiveness of text-mining techniques has been illustrated in a study by Bozorgi et al. (2010). They employ support vector machine (SVM) to predict time-to-exploit indicators of reported vulnerabilities on the Open Source Vulnerability Database (OSVDB) and *CVE*. They train a SVM classifier using the exploit-classification status reported in OSVDB as ground truth. Zhu and Dumitras (2016) apply NLP to extract malware detection features, from research papers automatically. The system they proposed, FeatureSmith specifically generates features for detecting Android malware through a semantic network. This system also mirrors human reasoning processes. Zhu and Dumitras (2018) also propose a ChainSmith system with multi-class classifiers to extract features from security articles for indicators of compromise (IOCs). Liao et al. (2016) systematically collect cyber-threat intelligence from technical blogs. From these technical blogs and reports, the authors mine IOC by matching them to the OpenIOC ontology through a rule-based extraction method. Yi et al. (2020) adapt an ensemble model to include rule-based, dictionary-based, and conditional random fields-based extractors, to further improve the performance of cybersecurity entity extraction. Scandariato et al. (2014) adapt bag-of-words for feature extraction, and employ Naïve Bayes and Random Forest classifiers for software-vulnerability classification and prediction. Bullough et al. (2017) evaluate performances of several prior ML models, including the SVM model from Bozorgi et al. (2010), to quantify the influence of class imbalance, as well as how training and testing datasets are divided. They use PoC exploits in ExploitDB as ground truths and extract training data from *NVD*. Kuppa, Aouad, and Le-Khac (2021) apply a multi-head joint embedding neural network model to map 62,000 *CVE* records to 37 ATT&CK techniques. The method they propose enriches *CVEs* with attack technique labels through NLP techniques and multi-label text classification models. Zhang, Caragea, and Ou (2011) use *NVD* data to predict the time until the next undiscovered vulnerability is reported in a particular piece of software. Bullough et al. (2017) evaluate performances of various ML models to generate predictions on vulnerability exploitation based on data from *NVD* and Twitter. They implement multiple ML models in their system to evaluate the influence of class imbalance and how training and testing datasets are divided. They also argue that using Twitter for vulnerability exploitability analysis is not practical.

Targeting *CVSS* base score generation, Gawron, Cheng, and Meinel (2017) apply Neural Networks and Naive Bayes algorithms. In contrast, Yamamoto, Miyamoto, and Nakayama (2015) deploy supervised LDA for *CVSS* metrics classification. Nevertheless, using correlated cybersecurity data sources also raises potential inconsistencies, such as the disparity between scores for the same vulnerability instances (Johnson et al., 2016b). One drawback of previous AI-based *CVSS* computing approaches is that they directly adopt the vulnerability reports and *CVSS* scores from *NVD* as training grounds, which may induce a bias in their model.

In comparison, we correlate vulnerability instances in *NVD* with the corresponding vendor reports and third-party cybersecurity analysts such as CERT reports to consolidate data sources. As a result, the relevant information is integrated into a unified struc-

Table 3.1: Text-mining approaches in cybersecurity analysis

Paper	Database	Model	Ontology	Domain
(Zhu and Dumitras, 2016)	Research papers	Semantic network; cognitive reasoning word2vec;	Drebin (App et al., 2014)	Malware Detection
(Zhu and Dumitras, 2018)	Research papers	multi-class classifier	N/A	Threat pattern
(Liao et al., 2016)	Industrial blogs	Dependency parsing; content term extraction; graph mining	OpenIOC	Threat pattern
(Husari et al., 2018)	Symantec Threat Reports	Part-Of-Speech tagging; TF-IDF	CAPEC and ATT&CK	Threat pattern
(Neuhans and Zimmermann, 2010)	CVE	LDA Topic Model	N/A	Vulnerability pattern
(Yi et al., 2020)	CVE; CERT; Email	Part-Of-Speech (POS); Ensemble extractor	N/A	Information retrieval
(Scandariato et al., 2014)	Fortify SCA reports	Naïve Bayes; Random Forest	N/A	Vulnerability prediction
(Bozorgi et al., 2010)	OSVDB; CVE	Bag-of-words; Support vector machines	Disclosure data; Exploit label in CVE	Vulnerability assessment
(Sabotthe, Suciu, and Dumitras, 2015)	Symantec Threat Reports; Twitter	Support vector machines	ExploitDB; OSVDB; Microsoft security advisories	Exploit prediction
(Bullough et al., 2017)	NVD; ExploitDB	TF-IDF; Support vector machines	N/A	Vulnerability assessment
(Almukaynizi et al., 2017)	ExploitDB; Zero Day Initiative; Dark-web	Support Vector Machine; Random Forest; Naïve Bayes;	CVSS; Symantec Threat Reports	Vulnerability prediction
(Zhang, Caragea, and Ou, 2011)	NVD; CPE	Logistic Regression; Multi-layer perceptron; RBF network; SMO regression; Gaussian processes	CVSS	Vulnerability assessment
(Christey and Martin, 2007)	CVE	N/A	CWE	Vulnerability pattern

ture as the training grounds. In doing so, our approach streamlines the computation of vulnerability severity to address such inconsistencies upstream to optimize security investments and shorten the potential risk window. Our proposed approaches utilize ML algorithms to streamline vulnerability assessments. We first initialize a vulnerability scoring system to rate the severity of a reported vulnerability instance, whereby standard *CVSS* metrics are used as a scoring basis to evaluate vulnerabilities' exploitability and the consequence of maliciously exploiting those vulnerabilities. This approach addresses compatibility issues across different *CVSS* versions, while automating vulnerability analysis. We also suggest a weakness categorization model that associates a higher abstraction level *CWE-ID* with a vulnerability report.

### 3.2.2 ENSEMBLE METHOD

The aforementioned ML-based works primarily focus on how to use ML techniques to make cybersecurity deductions and decisions in an automated manner, with limited contributions to the following aspects. Firstly, the availability of precise and conflict-free data sources is paramount considering security implications. Secondly, deployment of ML usage remains underutilized. The deployment of AI in various tools also contributes to the issues of interoperability and compatibility (Heelan, 2011). Nevertheless, existing ML-based methods are usually driven by specific cybersecurity problems such as detecting exploits. These customized approaches require specified ML algorithms and adjusted parameters to achieve optimized performances. Yet, a thorough vulnerability-driven cybersecurity assessment needs to consider various aspects such as vulnerability categorization, severity evaluation and exploit prediction. These diverse aspects require different learning models. Different models perform diversely for various tasks. For example, logistic regression (LR) (Almukaynizi et al., 2017) (Zhang, Caragea, and Ou, 2011) algorithm does not require heavy computing sources, but its prediction accuracy may be lower compared with more CPU-intensive neural network (NN) (Zhou et al., 2016) training. There is no one-size-fits-all solution that handles diverse classification and clustering tasks in a uniform manner (Torres, Comesaña, and Garc ía-Nieto, 2019) (Oprea et al., 2018). This is especially true in vulnerability analysis, considering the difficulty in the evaluation of related cybersecurity properties and the subsequent classification imbalance or bias (Sommer and Paxson, 2010).

Unlike a singular ML model, an ensemble method combines multiple ML models to produce one optimal predictive model (Tong, Liu, and Wang, 2018) (Lower and Zhan, 2020). Most ensemble approaches have built-in techniques to handle class imbalance. The effectiveness of ensemble ML techniques has been exemplified in the studies that are discussed later (Resende and Drummond, 2018). An example is a study carried out by Vanerio and Casas (2017) in which a supervised learning method named Super Learner is proposed to find the optimal ensemble model for anomaly detection. Li et al. (2017) utilize binary classification and k-nearest neighbor (KNN) classifiers into a hybrid intrusion detection solution. Similarly, Rajagopal, Kundapur, and Hareesha (2020) develop an ensemble paradigm based on meta-classification and stacking generalization, while also targeting prediction accuracy enhancement of network intrusions. Identifying base ML algorithms that can perform a security-focused classification task is crucial for constructing effective ensemble models that suit cybersecurity needs, such as improving the accuracy of attack detection (Seni and Elder, 2010) (Dietterich, 2000). For example, Fang et al. (2020) employ the LightGBM algorithm as their ensemble technique to predict exploits and track the footprints of vulnerabilities. These studies present wide applications of ensemble approaches, and thus reveal the potential of utilizing ensemble

techniques to improve general model performances such as prediction accuracy. However, there have been few attempts to use ensemble methods for multiple vulnerability-analysis purposes.

It is also critical to construct an appropriate combination scheme for selected base ML algorithms (Onan, Koruko\u0111lu, and Bulut, 2016). Common ensemble models combine several individual models by taking the average or a weighted average of these individual models. Some mechanisms for such an ensemble construction are voting (Kittler et al., 1998), bagging (Breiman, 1996), boosting (Freund and Schapire, 1995), stacking (Wolpert, 1992), mixture of experts (Waterhouse, MacKay, Robinson, et al., 1996), among others. A hard-voting scheme (or majority voting) is used only when the predicted labels are available. If continuous outputs such as posterior probabilities are accessible, a soft-voting scheme (or average voting) or other linear combinations may be adopted (Kittler et al., 1998). Stacking methods (Wolpert, 1992) are also used to train the output classifier, while taking feedback from input classifiers as new features. In bagging (Breiman, 1996) and boosting methods, homogeneous classifiers are usually trained using different samples of the dataset, to produce an ensemble model that is more robust than the individual classifiers. Both bagging and boosting mechanisms entail a large size of initial data to capture most of the complexity of the underlying distribution, in order to ensure that sampling from the dataset represents a good approximation of the real distribution. Some other construction mechanisms include adaptive ensemble models that handle non-stationary time series. For example, Larcher Jr and Barbosa (2019) apply adaptive ensemble methods to enhance the trade-off between cybersecurity requirements and system efficiency. Adaptive ensemble methods usually use weighted voting techniques that vary attribute weights across classifiers depending on selected metrics. Such a process can be further improved by taking into consideration outperforming metrics for more adaptive predictions (Quintal et al., 2020).

This thesis proposes an ensemble approach that combines independent classifiers while taking input data from heterogeneous cybersecurity sources on top of the vulnerability data curation process, which will be discussed later in Chapter 6 and the gap-filling ML mechanisms for CVSS and CWE categorizations. This approach yields better performance for cybersecurity classification tasks, while improving flexibility in handling diverse cybersecurity analysis missions. More specifically, we suggest an optimization algorithm that selects the best ML base algorithm(s) to construct effective ensemble models for diverse cybersecurity mission targets. Furthermore, a variant of cross-validation is adopted to minimize possible over-fitting across classification tasks (Van der Laan, Polley, and Hubbard, 2007). The optimization algorithm explores all combination schemes for selecting the best-performing ML-based instances.

### 3.3 SYSTEM-WIDE VULNERABILITY ANALYSIS

System-driven vulnerability assessment focuses primarily on the whole CI system, rather than on individual components (Kure, Islam, and Razzaque, 2018). Identifying individual vulnerabilities only is not sufficient or adequate in today's complex systems (Ammann, Wijesekera, and Kaushik, 2002). In system-driven techniques, chief security officers acquire the system's vulnerability score at the highest level. This is derived iteratively from the interdependent system components. Assessing the risk of a complex critical infrastructure, such as a power-grid system, necessitates the consideration of several vulnerabilities across highly interdependent CI components.

### 3.3.1 SEMANTIC MODEL

System-driven vulnerability assessment focuses primarily on the whole CI system, rather than individual components (Kure, Islam, and Razzaque, 2018). Identifying individual vulnerabilities only is not sufficient or adequate in today's complex systems (Ammann, Wijesekera, and Kaushik, 2002). In system-driven approaches, chief security officers obtain a system's high-level vulnerability score. This is derived iteratively from the system parts that interact with each other. Risk assessment of a complex critical infrastructure like a power-grid system involves various vulnerabilities across highly interdependent CI components.

Risk assessment of a complex CI such as a power-grid system involves analyzing various vulnerabilities across highly interdependent IT and OT components. Identifying only individual vulnerabilities and threats is not sufficient in today's complex systems (Kure, Islam, and Razzaque, 2018). Different modeling attempts have been made to pinpoint both individual vulnerabilities (e.g., legacy software) and structural vulnerabilities (e.g., lack of network segmentation) (Blockley et al., 2002). Data visualization models such as tree structures, directed graphs, and logic diagrams are widely used for system-wide cybersecurity assessment or exploit modeling (Noel et al., 2016) (Lallie, DeBattista, and Bal, 2018). However, many of the previous tree structures or graph-based studies are usually ad-hoc to specific system structures or network environments to evaluate the likelihood or the possible consequence of exploiting specific vulnerabilities, such as DoS and Man-in-the-Middle (MiTM). Flexibility and extensibility are usually not the prime designing criteria (Noel et al., 2016). In other words, existing cybersecurity assessment frameworks may require substantial reconstruction to validate a different type of vulnerability, and are therefore neither effective nor economical. Moreover, the experimental datasets or evaluation datasets are mostly not published, which makes the process hard to reproduce (Eckhart and Ekelhart, 2018).

Many valuable frameworks are proposed to address increasing security issues in the OT systems, including NIST SP 800-82 for the ICS security (Stouffer, Falco, Scarfone, et al., 2011), NIST Cyber Security Framework for Improving Critical Infrastructure Cybersecurity (2014), and *North American Electric Reliability Corporation Critical Infrastructure Protection* (2008) standards. In addition, several international standards specifically focus on security in the domain of the smart grid, such as IEC 62351 (entitled "*Power systems management and associated information exchange. Data and communications security*") and NISTIR 7628 Rev. 1 (entitled "*Guidelines for Smart Grid Cyber Security*"), which are summarized by Ruland et al. (2017). Due to the IT and OT convergence introduced in the previous section, the scientific community and the industry continue to search for solutions to bridge the gaps between IT and OT security. However, (Conklin, 2016) suggests that the adaption of IT-specific security regulations(e.g., NIST SP 800-53) in OT security directives (e.g., NIST SP 800-82) leaves the fundamental business objective differences between IT and OT systems unaddressed.

Risk modeling languages, such as semantic maps and ontologies, for model-based security engineering have been proven to be scalable and flexible (Nguyen, Ali, and Yue, 2017) (Zhou et al., 2012). Several enterprise architecture frameworks have been developed to support risk presentation and analysis, such as CORAS by Fredriksen et al. (2002). There are also newer works such as Secure-i\* by Liu, Eric, and Mylopoulos (2009) and Secure-Tropos by Mouratidis and Giorgini (2007). These semantic works are beneficial, yet the most of them are overly generic. Thus, these works rely on the proficiency of their consumers. In order to conduct an effective cybersecurity analysis, ontologies developed for cybersecurity reasons and CI operations must be combined.

There have been few attempts to create a single ontology for IT/OT security in CIs, as the varied descriptive terminology given in these two domains result in considerable heterogeneity and little interoperability. Some works (e.g., (Venkata, Kamongi, and Kavi, 2018) (Mozzaquattro et al., 2018)) connect *Common Platform Enumeration (CPE)* (2022) ontology with vulnerability databases, meaning that vulnerability information for different components can be integrated into their ontology. In these works, ontologies are applied to provide a formal and explicit way to specify concepts and relationships. In the study by Venkata, Kamongi, and Kavi (2018), for instance, public vulnerability data seeds from repositories such as *Common Platform Enumeration (CPE)* (2022) (CVE and CPE are correlated to their ontology knowledge base, and further mapped through the STRIDE (Khan et al., 2017) threat categorization. However, this work does not consider further reasoning and logical analysis of how its ontology correlates with various vulnerabilities, threats, and mitigations. Mozzaquattro et al. (2016) propose an IoTSec ontology-based framework that combines both model-driven development and ontology-driven development. This framework covers two use-case scenarios, i.e., one for design purposes and the other for run-time system security monitoring and management. Mozzaquattro et al. (2018) further extend the IoTSec reference ontology into a database of IoT cybersecurity knowledge (vulnerability, threat, and prevention mechanism) to support cybersecurity analysis.

Enterprise modeling has also been proposed for supporting cybersecurity management by a number of authors. Pavleska et al. (2019) develop a reference architecture for evaluating information security of an enterprise architecture. Their conceptual framework covers security goals, vulnerabilities, threats, and security measures, all being linked to the enterprise model. The reference architecture is a guideline to manually assess the security status of an enterprise, using its enterprise model. Grandry, Feltus, and Dubois (2013) extend the ArchiMate meta-model by cyber security concepts such as risks, threats, vulnerabilities, security goals, and countermeasures. Efforts to include security aspects into ArchiMate (Ellerm and Morales-Trujillo, 2020) predominantly focus on design rather than analysis of vulnerabilities. ArchiMate is not (yet) designed to cover the plethora of OT and physical components found in CIs like smart grids.

Our taxonomy is based on existing models and our own CI investigations. When establishing vulnerability attributes for our CI entities, for instance, we adopted and improved upon the CVRF framework (Schiffman, 2011) to make cybersecurity information exchange with large security alert repositories such as NVD more accessible. In doing so, we propose a taxonomy that includes IT and OT components, their attributes such as potential vulnerabilities, and their linkages, which can serve as the foundation for IT/OT convergence research. Then, we construct power-grid reference models that demonstrate the applicability of our taxonomy in bridging terminologies used in the IT and OT cybersecurity domains to improve situational awareness of CI cybersecurity.

### 3.3.2 REFERENCE ARCHITECTURE

Reference models are widely used in system modeling or model-based system engineering to support security-driven analysis (Cloutier et al., 2010). They capture the typical topological structure and functional relationships of designs. By generalizing these common traits, they promote reusability. In the smart grid study and its sub-field, reference model research has been conducted to provide formal and clear guidelines for the design of power-grid architectures. The IEEE P2030 Smart Grid Interoperability Framework (Photovoltaics and Storage, 2011), the EU Mandate M490 SGAM (Gottschalk, Uslar, and Delfs, 2017), and others are examples of national or international initiatives to standard-

ize the smart grid.

Some studies attempt to enhance both abstraction and extensibility of the power-grid reference architecture through ontology or meta-modeling. Irlbeck et al. (2013) propose a bottom-up reference architecture for smart grid and discusses the problems and goals associated with establishing such designs in Europe. Bytschkow et al. (2014) present a CPS reference framework, which is then used to describe cross-domain relationships in the smart grid and automotive sectors. Korman et al. (2016) offer a reference architecture to combine advanced metering infrastructure and cybersecurity research, with the reference model serving as an instance of their suggested meta-model. Their smart metering reference model conforms to UML syntax and provides automated EA (enterprise architecture) analysis via OCL or P2AMF implementation. They indicate that a reference model alone cannot satisfy all needs, such as availability, uncertainty, and the expression of validation constraints. Reference models in conjunction with a modeling tool can instead meet these needs at a higher level. They further improve their smart-grid reference model to provide functional and data-flow-oriented reference architecture models to automate security evaluations and cyber-attack simulations. Their work focuses mostly on the cyber network model for smart metering and related load balancing capabilities. The European SEGRID project (SEGRID Consortium, 2017) also provide significant reference models for smart grids, but focuses solely on communication and enterprise modeling while ignoring the physical components. It also offers minimal information on the controlling network and other components.

The majority of current power-grid reference models aim to provide architectural snapshots. However, adaptability and extension of the system structure may not be the primary design objective. However, the reusability of a reference design is essential for allowing model updates that bring the suggested architecture up to date. The proposed cyber-physical reference models for the power grid offer reusability and efficiency with standardized virtual replicas for cyber connections, cyber-physical configuration, and physical operations. Using the proposed taxonomy, the instantiated reference models can be easily maintained and adjusted. In the meanwhile, we considered noteworthy contributions from smart-grid-related documents such as the IEC 62351 series.

### 3.3.3 CYBER-PHYSICAL DEPENDENCE ANALYSIS

A complex CI is a SoS that integrates a collection of devices to achieve the desired capabilities (Uslar et al., 2019). In addition, there are complex interaction dependencies between interconnected components (Kong, 2019). The dependencies in such a SoS are divided into inter- and intra-dependencies. The inter-dependencies and intra-dependencies of CIs such as smart grids implicitly determine the cascading effects and the system resilience under potential attacks or failures (Marashi, Sarvestani, and Hurson, 2017).

Akbarzadeh and Katsikas (2021) suggest an application of modeling and simulation methods to study CPSs and detect dependency chains. They also provide an approach to identifying and analyzing inter-dependencies and intra-dependencies between subsystems of a complex system by quantitative measures of the impact of dependency, susceptibility of dependency, and weight of dependency. Besides the study by Akbarzadeh and Katsikas (2021), valuable research works have been carried out for modeling dependencies in CIs in terms of cybersecurity enhancement of such complex systems (Chopade and Bikdash, 2011). Ouyang (2014) reviews six significant types of approaches for modeling interdependencies among CIs, such as empirical approaches and agent-based approaches, and suggested the necessity of an open modeling framework to allow adjust-

ment of CI models. König et al. (2019) propose a combination of local and global views and illustrates the common practical division of the physical and cyber domains. They use a limited number of data elements, including assets, interdependencies, and linkages between assets, as well as events and alerts linked with assets. The physical and cyber parts require these items of the system. They also provide a high-level description of how these parts inter-operate, which expands awareness from “knowing what” to “knowing what will happen next”, thus solving the core responsibilities of effective risk management. Kwasinski (2020) study the network and physics of the power grid dependence within the domain and confirms the cyber-physical properties of the power grid. This study shows that internal dependence reduces the resilience of the power system, while service buffers (such as energy storage or data connection re-establishment wait times) help to limit the impact of internal dependencies on resilience. Therefore, understanding and discovery of internal cyber-physical dependencies are essential to the security analysis of complex CPSs (Chopade and Bikdash, 2011). Actually, lower resilience in the cyber-domain vertices is more critical than lower resilience in physical domain vertices (Kong, 2019).

In this thesis, we modeled intra-dependencies within CIs in order to evaluate the interactions between cyber networks and physical controlling networks. However, inter-dependencies between connected CIs, such as smart grids and water distribution systems, are beyond the focus of this research. These intra-dependencies are multidimensional and are classified as functional, logical, geographical, social, and economic connections. Functional dependence means that a task of one component is functionally dependent on another component (Wang, Xing, and Levitin, 2012) (Zhao and Xing, 2019). Logical dependence is an implied relationship between two components, which is commonly observed in software development (Oliva et al., 2011). Spatial dependence is the propensity for two close components to have a greater likelihood of influencing one another. Two physical servers housed in the same office have an increased risk of fire propagation, which is a typical example of spatial dependence. In addition, two software components embedded in the same hardware are interdependent due to the competition between computing sources. Social dependence implies the influence of social elements such as energy sector policies. In contrast, economic dependency is related to expenditures or earnings, such as business competitiveness. Social-economic dependence also encompasses instances in which various entities collaborate and are responsible for different areas of the smart grid (Palm, 2021). This thesis is only concerned with functional dependencies, namely cyber- and cyber-physical functional dependencies. Physical dependencies such as electricity generation, transmission, and distribution are not explored in this thesis.

Seven functional dependence rules are defined and expanded in Chapter 8. These functional dependence rules bolster modeling of cascades and criticality analysis. The position of a component inside a network system differentiates the component’s significance, and consequently leads to varying degrees of system failure. Zhu and Milanović (2017) suggest a way for weighted modeling CPS in which a three-dimensional weighted complex network model is introduced. In heterogeneous systems, the various engineering structures can be modeled without modifying the topological model. Due to the inter-dependencies across systems, the sophisticated network-based models expose the susceptibility of various engineering systems and the essential components that could trigger a cascading failure. Myhre et al. (2020) evaluate the betweenness centrality of the components in a combined electrical grid and ICT system using complex network theory. Additionally, they model the impact on the system when individual nodes are removed in order to further diagnose crucial nodes. The propagation of defects from the network to the physical device will have the most negative impact on the system’s re-

liability. Marashi, Sarvestani, and Hurson (2017) offer an analytical reliability model that takes into account the consequences of damage to physical and cybe-components, as well as the effects of cyber-physical linkages between these components.

# RESEARCH METHODS



# CHAPTER 4

## RESEARCH METHODS

This chapter explains the research paradigm and methods utilized throughout the thesis. Then, a description of the research process is provided to map the proposed research methodologies to the various research stages. The evaluation methodologies and metrics used for each study strategy are also addressed.

### 4.1 RESEARCH PARADIGM AND METHODOLOGY

Andersen and Hepburn (2016) characterize scientific activities as “*systematic observation and experimentation, inductive and deductive reasoning, and the formation and testing of hypotheses and theories*”. Scientific methods are the means by which these activities are conducted to establish facts and reach new conclusions. Saunders, Lewis, and Thornhill (2009a) compares the research process to the layer-by-layer uncovering of an onion. This approach depicts a sequence of stages via which various data collection methods can be comprehended and defined. This thesis employs the research onion model to explain the research paradigm, methodology, and tools methodically. Following this methodology, the selected study design of this thesis is marked in red, as seen in Figure 4.1. Robson and McCartan (2016) emphasize that research should be conducted methodically and ethically. Next, the adopted research paradigm is described to demonstrate the methodical and ethical nature of the proposed works.

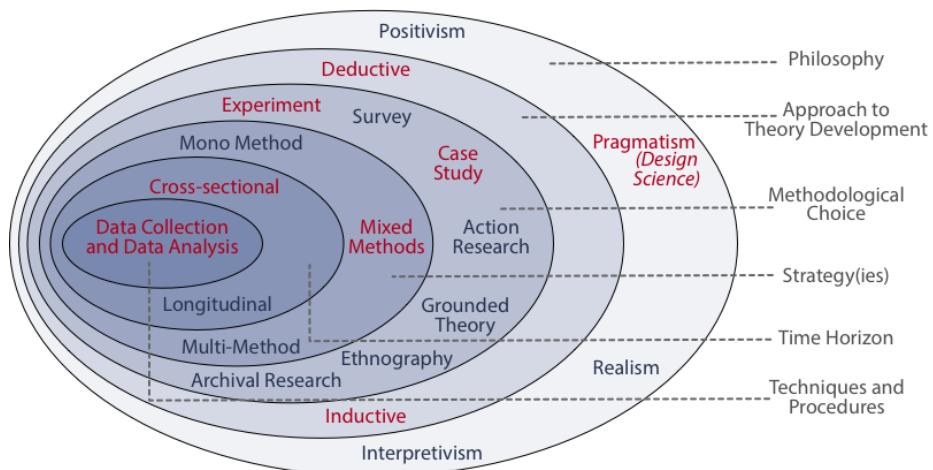


Figure 4.1: Research methodology adapted from the research onion model of Saunders, Lewis, and Thornhill (2009a)

### 4.1.1 DESIGN SCIENCE RESEARCH PARADIGM

A research paradigm is a fundamental set of common beliefs and agreements shared between scientists about how research problems should be understood and addressed, and how knowledge is gained or created (Kuhn, 2012). A research paradigm comprises awareness of different theories and practices used to carry out research in a particular discipline (Creswell and Clark, 2017). Typically there are four main types of paradigms which are positivism, interpretivism, critical realism, and pragmatism. Positivism and interpretivism are the two major traditional trends adopted in quantitative and qualitative researches, separately (Robson and McCartan, 2016). These two paradigms are mutually exclusive as positivism is more frequently used in quantitative works that are independent from human interpretations, while interpretivism is more commonly used in qualitative works where human interpretations are studied. Critical realism and pragmatism are related to either positivism and interpretivism. Design science research (DSR) paradigm is commonly employed within the information system. Hevner (2007) claims that "*design science research is essentially pragmatic in nature due to its emphasis on relevance; making a clear contribution into the application environment*". This paradigm can be viewed as a theoretical construct that describes and explains technological development (Ropohl, 1999). In this perspective, science and technology are both indispensable sources of investigation.

According to Oates (2005) and Hevner et al. (2004), a philosophical paradigm can be characterized using four components, namely ontology, epistemology, methodology and axiology. Ontology is concerned with assumptions about the form and nature of reality. And hence, ontological assumptions shape how a researcher perceives and studies research objects. Epistemology refers to the nature of knowledge and also the ways how knowledge is constituted and transferred (Saunders, Lewis, and Thornhill, 2009b). Epistemological assumptions define what is acceptable, valid, and legitimate knowledge. Axiological assumptions represent a researcher's vision of the role of values. The characteristics of the DSR paradigm are further summarized in Table 4.1 using these four metrics.

Table 4.1: Design science research paradigm adapted from Saunders, Lewis, and Thornhill (2009b) and Vijay, Bill, and Stacie (2015)

Paradigm	Ontology	Epistemology	Axiology	Approach
Design Science	Multiple meanings; Depending on the technological and social development.	Knowledge is created through setting objectives and iterations; Contribution may include future practices or artifacts.	Value-driven; Researcher creates, designs, improves and understands.	Developmental/practical solutions; Technological augmentations to social and individual factors.

A scientific inquiry may take an inductive, deductive, or abduction form (Flick, 2015). The goal of an inductive research is to generalize and infer theoretical conclusions from observed data. Whereas, the goal of a deductive research is to test and validate a theoretical hypothesis based on a pre-existing theory considering new data. Abduction, on the other hand, provides the logic to discover grounded theories related to the understanding of unstructured phenomena. A DSR paradigm may employ combinations of inductive, deductive and abduction reasoning during the research process.

#### 4.1.2 DESIGN SCIENCE METHODOLOGY

A research methodology describes the general principles to guide research activities (Oates, 2005). The research question and the investigated phenomenon determine what methodology can answer the research question most effectively and efficiently. Cybersecurity is a socio-technical discipline, as introduced earlier in Section 1.7.3. The research problem and the questions addressed in this thesis require a rich understanding of the current socio-technical developments in cybersecurity analysis and management, including technical evolutions and social actors' involvement. This thesis focuses on the socio-technical properties of vulnerability-analysis processes in an iterative to accommodate new requirements. The far-reaching contribution of this thesis leads to a set of new methods with embedded models and algorithms to support data-driven vulnerability analysis in complex CIs. Therefore, the DSR paradigm was considered the most appropriate approach to elaborate on the proposed methods.

A research strategy refers to how to combine research methods, and includes mono, mixed, or multi-strategy (Saunders, Lewis, and Thornhill, 2009b). Mono strategy means that only quantitative or only qualitative methodology is chosen. Mixed strategy refers to combinational methodology with both quantitative and qualitative data are collected. Multi-strategy is similar to the mixed method in the sense that both quantitative and qualitative methodologies exist in the research, but differs in that only one type of data is collected. In the field of cybersecurity, social sciences and technical sciences play equally important roles (Trist, 1981). Therefore, cybersecurity research relies upon observations from both sides to form coherent explanations (Spring, Moore, and Pym, 2017).

According to Simon (2019), DSR originated from the science of the artificial which is centered around human-made artifacts that meet some actual needs. Hevner and Chatterjee (2010) also suggest that DSR “*is a research paradigm in which a designer answers questions relevant to human problems via the creation of innovative artifacts, thereby contributing new knowledge to the body of scientific evidence. The designed artifacts are both useful and fundamental in understanding that problem*”. DSR employs a combination of different facts (social, behavioral, and formal) to explore, explain and evaluate actions for the development and future use of artifacts. Such artifact can be a *construct*, a *model*, a *framework*, an *architecture*, a *method*, or an *instantiation* (Vijay, Bill, and Stacie, 2015). For example, a construct contains various conceptual elements, while a model describes the relationships between these elements. A framework represents the essential structure of the artifact components. A method refers to a guideline, an algorithm, or a practice to realize the artifact. An instantiation is considered as the implementation of the artifact in a specified context.

Cybersecurity studies focus on developing such artifacts that provide desired computational functions without being vulnerable to relatively trivial malicious attacks (Landwehr, 2012). Particularly, this thesis contributes to the knowledge of how to perform vulnerability analysis for complex CIs with the support of publicly accessible vulnerability repositories, by constructing four complementary artifacts (i.e., a vulnerability data model, a ML-based vulnerability analysis method, a CI vulnerability taxonomy, and a vulnerability analysis streamlining method) as well as their instantiations, as introduced earlier in Section 1.5. In doing so, this thesis enhances the descriptive, prescriptive, and predictive advances in CI vulnerability analysis practices, as shown in Figure 1.2 in Section 1.4.

Under DSR paradigm, further methodology-related concepts are introduced next to understand their usage within the context of this thesis.

### 4.1.3 RESEARCH METHODS

Gregor (2006) classifies theories for IS into theories for analysis, explanation, prediction, explanation, and prediction, as well as design and action. This thesis utilizes various types of theories during different stages of the research process shown in 4.3. More specifically, theories of explanation are taken into consideration at the initial stage of research problem awareness to explore the challenges of CI vulnerability analysis using publicly accessible vulnerability repositories. In addition, theories of design and action are applied for the design, demonstration, and evaluation of the proposed artifacts to better measure their utility. Here we present the definitions of these two types of theories that further influence the selection of research methods:

*“A theory of explanation says what is how, why, when, and where. The theory provides explanations but does not aim to predict with any precision. There are no testable propositions.*

*A theory of design and action says how to do something. The theory gives explicit prescriptions (e.g., methods, techniques, principles of form and function) for constructing an artifact.”*

— Gregor (2006)

A research method depicts how a research task is explored, described, explained, or performed using specific techniques and tools. Hevner et al. (2004) introduce five types of research methods to evaluate knowledge generated from DSR processes, namely observational, analytical, experimental, testing, and descriptive methods. Next, we discuss these five different methods.

- Observational methods include case studies or field studies that are carried out to learn the usage of an artifact in a specified context (Yin, 2011). Field study, on the other hand, involves actual workplace settings for observing, interacting, and understanding stakeholders while they are immersed in their natural environment.
- Analytical methods include static analysis, architecture analysis, optimization, and dynamic analysis, which can help to observe how variables of interest related to other factors. The static and dynamic analysis methods examine the structure of the artifact to analyze the static and dynamic qualities, separately. The architecture analysis method studies the fit of the artifact into technical IS architecture.
- Experimental evaluation strategies include laboratory and field experiments, as well as computer and experimental simulations. When applying experimental methods, the researcher manipulates some factor controls and measures any change in other factors (Berger and Maurer, 2002), or conducts simulations with artificial data.
- Testing methods include both functional and structural testing methods. The formal one also refers to black-box testing that discovers failures and defects. The latter method (white box) covers the approaches for performing coverage of testing in artifact implementation.
- Descriptive methods include informed argument or scenario descriptions. The informed argument suggests that information from the knowledge base, such as relevant research results, are provided as convincing arguments. The latter, or scenario descriptions, construct specific scenarios to demonstrate the utility of an artifact.

Table 4.2 summarizes various research methods during different stages in this thesis.

Table 4.2: Adopted research methods

Activity	Method	Data Collection	RQ
Study I on baseline analysis	Observational study	Publicly accessible repositories Literature review	RQ (1)
Study II on users' perspectives	Survey	Online questionnaire	RQ (1)
Study III on data quality impact	Case study (exploratory)	System documentation Interview	RQ (1)
Study IV on real-world vulnerability analysis using <i>Artifact-I</i>	Archival analysis	Publicly accessible repositories	RQ (2)
Study V on comparative analysis	Static analysis	Contemporary practices review	RQ (2)
Study VI on <i>Artifact-II</i> CVSS categorization method performance	Experiment	Publicly accessible repositories Literature review	RQ (3)
Study VII on real-world vulnerability severity scoring using <i>Artifact-II</i>	Archival analysis	Publicly accessible repositories	RQ (3)
Study VIII on <i>Artifact-II</i> CWE categorization method performance	Experiment	Publicly accessible repositories Literature review	RQ (3)
Study IX on <i>Artifact-II</i> ensemble ML method performance.	Experiment	Publicly accessible repositories Literature review	RQ (3)
Study X on power-grid modeling and analysis using <i>Artifact-III</i>	Simulation	Literature review Case study	RQ (4)
Study XI on analyzing manufacturing vulnerability using <i>Artifact-IV</i>	Case study (explanatory)	Literature review System documentation Interview	RQ (4)

According to Hevner and Chatterjee (2010), simulation can be used to “*better understand the original (simulated) entity because simulation can help predict behavior by making explicit “new” knowledge, i.e., the knowledge that is indeed derivable but only with great effort*”. Due to privacy and security concerns, security experiments in actual systems are limited and might expose these systems to potential threats. Instead, simulation-based methods contribute as both experiment base and training exercise tools for security professionals while bringing minimal potential consequences to the real world (Philosophy, 2013) (Veksler et al., 2018) (Benzel, 2011). Comparative experiment-based evaluations are carried out to assess the quality of *Artifact II*, with evaluation criteria drawn from ML literature.

Network models of infrastructures are employed for attack scenarios or user-defined vulnerability simulations. A simulator records the states of a system at time  $t, t+1, t+2, \dots$ , to produce a numerical picture of the system’s evolution under specified variables. Such simulations enable CI vulnerability assessment using *Artifact III*. Evaluation criteria are drawn from the ontology literature.

We consider the case study definition from Runeson et al. (2012) that is based on the original definition from Yin (2009). Runeson et al. (2012) states that a case study in software engineering serves as “*an empirical inquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the*

*boundary between phenomenon and context cannot be clearly specified*”. Although the studies performed in this thesis are not necessary for the field of software engineering, this definition still provides valuable references. Case-study methods are adopted for problem awareness (i.e., *Study III*) in the initial stage as well as evaluations (i.e., *Study XI*) of *Artifact IV*.

Two more archival analysis studies (i.e., *Study IV* and *Study VII*) are performed to evaluate the usefulness of *Artifact I* and *Artifact II* on vulnerability analysis using historical vulnerability instances retrieved from publicly accessible repositories. Here, archival analysis is defined by Wohlin (2021) that meet: “*(1) Evaluations or illustrations on a limited scale, for example, in a laboratory. These could be labeled evaluations or illustrations given their primary objective, and (2) Studies of existing information in open source repositories, defect databases of other database sources*”.

Meanwhile, a static analysis approach is also used in the comparative *Study V* to evaluate utility of *Artifact I*, with criteria drawn from database studies.

#### 4.1.4 DATA GENERATION METHODS

In this thesis, both quantitative data and qualitative data are collected to support vulnerability assessments. Quantitative data is further composed of: (i) vulnerability scores retrieved from online cybersecurity repositories to quantify vulnerability severity; (ii) component criticality based on functional and structural dependencies. Qualitative data is further composed of three subsets: (i) CI system-related information such as CI-component types, names, and connections are gathered through observations, documentations, and expert interviews, to support CI modeling; (ii) Users’ views on using open-source cybersecurity repositories in their security practices, which is gathered through the questionnaire-based survey; and (iii) Experts’ feedback on study results of vulnerability assessment in an actual large data center, which is collected through interviews. And hence, a mixed strategy is employed in this thesis, as shown in Table 4.2.

Data collected from running these simulations measures how different defense mechanisms and various vulnerabilities affect system-wide security. And then, user-defined vulnerabilities are described and coded in proportion to online vulnerability records. This step ensures that the employed simulation studies reflect both the latest vulnerability occurrences and also domain-specific vulnerabilities.

## 4.2 DESIGN SCIENCE RESEARCH PROCESS

This thesis employs five DSR processes to develop and demonstrate artifacts that bridge the knowledge gaps among cybersecurity functions and orchestrate cybersecurity activities for complex CIs. These five research processes, corresponding artifacts, and the connections among different processes are illustrated in Figure 4.2.

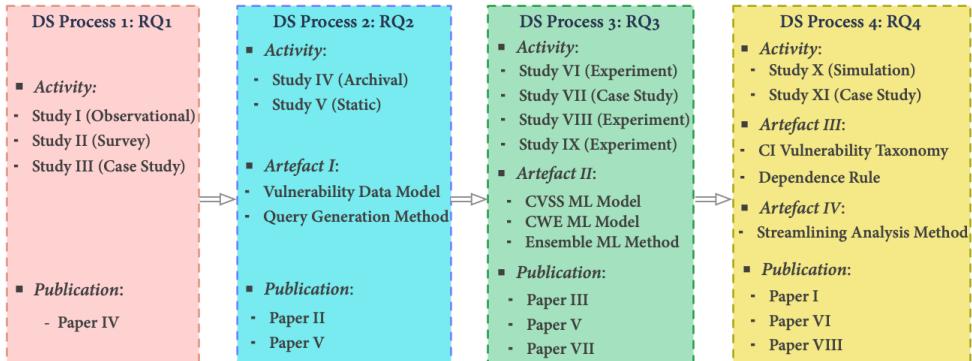


Figure 4.2: Design science research process

In this thesis, each research process is conducted following the framework depicted in Figure 4.3. This framework mainly adapts DSR-process structures provided by Hevner and Chatterjee (2010), while taking into consideration of the framework for the production and presentation of DSR in IS by Peffers et al. (2007), and is also inspired by Johannesson and Persson (2014).

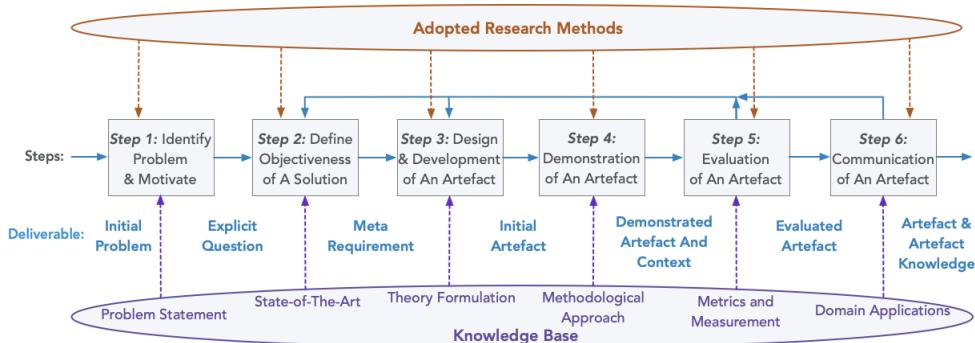


Figure 4.3: Design science research framework adapted from Hevner and Chatterjee (2010) and Peffers et al. (2007)

Mainly six steps are included in the process. An initial research problem is converted into explicit research questions that target particular objectives and lead to meta requirements. Accordingly, an artifact is designed, developed, demonstrated, evaluated, and communicated to the related scientific forum. The evaluation and communication of the artifact may generate feedback that is used to better its design and development. The DSR research process is therefore iterative, with theory and observation in constant interaction. The research methods at the top and the dashed arrows represent specific tools and approaches that assist the DSR process's tasks. The knowledge-base circle and dashed arrows show the scientific processes and foundations used to design, produce, and assess the artifact. The foundations include existing theories, models, methods, and experiences that this DSR could build upon.

#### 4.2.1 RESEARCH PROCESS 1: EXPLORATORY STUDIES

This research process starts with data collections from online cybersecurity data sources like *NVD*, with which a baseline study is performed to investigate how to access and interpret such data, as well as the patterns of historical records disclosed in *NVD* considering their correlations with standards such as *CWE* and *CAPEC*. This baseline study (i.e., *Study I*) lays the foundations and motivations for the following research processes, by trying to identify the challenges of applying cybersecurity repositories to support CI vulnerability assessment.

As part of this research process, a survey was conducted to gather data about industrial and academic perceptions of CI cybersecurity in terms of support from publicly accessible cybersecurity repositories and enumerations. Around 1200 surveys are sent out to stakeholders, including cybersecurity professionals as well as practitioners working in the area of CI cybersecurity domains, with monitoring and operational related responsibilities as well as academics working in related fields. The survey results of the received 410 responses are discussed in Section 5.3.

The baseline and survey studies gather the details about the phenomenon under investigation. These two studies also set up a hypothesis that the issues of heterogeneity, diversity, incompleteness, and inconsistencies of public vulnerability repositories pose obstacles to their applications in supporting CI vulnerability assessment. We carried out a case study (i.e., *Study III*) to validate this hypothesis, which specifically looks into the impact of data incompleteness and inconsistencies of these repositories. This case study was performed with actual system information (including system configuration and component information) from an actual data center. We also interviewed security professionals working in this data center to collect their feedback on the case study results, as shown in Table 4.3.

Table 4.3: Research process 1 - exploratory studies

Adopted Methods	Deliverable	Supporting Knowledge
Step 1: literature review.	Initial Problem: complex CI vulnerability assessment processes that involve multiple vulnerability data-sources.	CI cybersecurity.
Step 2: literature review.	Research Question: what are the challenges in CI vulnerability assessment using publicly accessible cybersecurity repositories?	Cybersecurity domain knowledge; Vulnerability-analysis lifecycle; Online cybersecurity data.
Step 3: <i>Study I</i> .	Purpose of Study: to identify how to access and interpret cybersecurity data sources.	Online cybersecurity data; Statistical analysis.
Step 4: <i>Study II</i> .	Purpose of Study: to understand the challenges in security-repository deployment from a user's perspective.	Qualitative analysis; Survey study.
Step 5: <i>Study III</i> .	Purpose of Study: to investigate the impact of security repository data quality on vulnerability assessment.	CI cybersecurity analysis; Data quality analysis; CI system characteristics.
Step 6: publication.	Paper IV.	Cybersecurity research community.

## 4.2.2 RESEARCH PROCESS 2: VULNERABILITY DATA CURATION

The industrial survey presented in *Research Process 1* emphasized the issues of heterogeneity and incompleteness of online vulnerability data sources and the problems induced by the static vulnerability-analysis lifecycle. These challenges lead to the research question in *Research Process 2*: how to obtain and correlate data for vulnerability analysis considering complex and heterogeneous sources among security alert repositories, considering that most vulnerability reports are written in a natural language?

Table 4.4: Research process 2 - vulnerability data curation

Adopted Methods	Deliverables	Supporting Knowledge
Step 1: industrial survey.	Initial Problem: heterogeneous, incomplete and redundant vulnerability data-sources; static vulnerability-analysis lifecycle.	Vulnerability-analysis domain.
Step 2: literature review.	Research Question: how to obtain vulnerability reports written in natural language considering heterogeneous sources of security alerts?	Cybersecurity domain knowledge; Vulnerability-analysis lifecycle; Online cybersecurity data.
Step 3: contemporary practices review.	Requirement Specification: correlates separated cybersecurity repositories, to enable context-aware data analysis that aids cybersecurity situation-awareness.	Existing data-correlation works; Vulnerability-analysis lifecycle.
Step 4: cybersecurity database specification.	Artifact I: a vulnerability common data model and related data correlation methods.	Cybersecurity data-repositories; Database construction; Correlation and reasoning.
Step 5: localized database instantiation.	Correlating 12 ( <i>CVE</i> , <i>NVD</i> , <i>CWE</i> , <i>CAPEC</i> , <i>Red Hat</i> , <i>MITRE</i> , <i>cvedetails</i> , <i>SecurityFocus</i> , <i>Shodan</i> , <i>ATT&amp;CK</i> , Manufacturer, and third-party analyzer (e.g., <i>ICS-CERT</i> )) repositories for CI vulnerability analysis.	Cybersecurity data-repositories; Database construction; Document-based database.
Step 6: <i>Study IV</i> for utility evaluation.	Evaluated Artifact: is applicable for CI vulnerability-characteristic analysis.	CI cybersecurity analysis; Statistical analysis.
Step 7: <i>Study V</i> for performance evaluation.	Evaluated Artifact: performance trade-offs compared to similar works.	Vulnerability-driven CI cybersecurity analysis; Database quality assessment.
Step 8: publication.	Artifact knowledge: communicated through Paper II and V.	Cybersecurity research community and stakeholders.

Table 4.4 summarizes performed studies to address these issues. More specifically, we correlate numerous widely used vulnerability data sources to incorporate distinct viewpoints from security repositories, motivated by related works summarized in sub-section 3.1.1. In addition, we define a common data model and related correlation algorithms, which are implemented as a localized and synchronized database to facilitate vulnerability analytics. This instantiated database artifact is validated through two case studies. One study (i.e., *Study IV*) explores the applicability of the proposed vulnerability database (containing 12 data sources) in the context of CI vulnerability analysis to gain insights into the threat landscape in CI environments. Then, a comparative analysis

study (i.e., *Study V*) examines the utility of the proposed approach against current relevant works, including standalone repositories (e.g., *CVE*) as well as similar correlation solutions (e.g., *CVE-Search* and *Security Database*).

#### 4.2.3 RESEARCH PROCESS 3: BRIDGING THE DATA GAP

The research process answers how can manual procedures of severity assessment and classification of weaknesses be supported by ML algorithms, and is summarized in Table 4.5. The literature review results of this process is summarized in Section 3.2. We proposed three sub artifacts to answer our research question, and following ML classification evaluation metrics presented in sub-section 7.2.3.

*Artifact II-A*, or the ML-based vulnerability scorer, is evaluated through an experimental study (i.e., *Study VI*) that contrasts our solution against several similar works. This experimental study involves two groups of comparison studies. Firstly, the performances of two sets of models are assessed, one set that only uses *NVD* as the data source for training data. In contrast, the other set uses both *NVD* and *SecurityFocus* as the training data sources. Secondly, the performance of the LR-based model is compared with the models from two related studies in terms of classification accuracy. This artifact also proves better performances with similar models from related works, and is also validated through a case study (*Study VII*) on CI vulnerability assessment.

*Artifact II-B*, or the ML-based weakness classifier, is evaluated through an experimental analysis (i.e., *Study VIII*). The proposed solution is initialized through a bidirectional long-short-term-memory (LSTM) neural network (NN) model and global vectors for word representation (*GloVe*) embedding. The validation is done by comparing the solution performance with existing weakness categorization and prediction studies.

*Study VII* explores the applicability of the proposed vulnerability-severity scoring mechanism in CI vulnerability analysis. Four CI asset types, namely RTU, MTU, PLC and HMI, are selected for this case study. The statistical exploitability, scope, and impact characteristics of CI vulnerabilities are presented in sub-section 7.3.5.

Experimental studies for these two initial artifacts show that the adoption of ML techniques in the previously human-centered vulnerability-analysis process contributes to higher automation in such processes and higher accuracy compared to manual vulnerability assessment in terms of severity scoring and threat categorization.

We diagnosed one limitation in these two initial artifacts that ML techniques are still underutilized in vulnerability assessment of diverse dimensions. To address the issue that there is no one-size-fits-all solution for various vulnerability assessment aspects, we develop a selective ensemble model on top of the ML-based vulnerability scoring. This artifact (i.e., *Artifact II-C*) or the selective ensemble model is evaluated by a comparative experimental investigation (i.e., *Study IX*). This is to demonstrate how the proposed ensemble model outperforms individual ML models. This study compares the suggested approach against a number of prevalent text-mining approaches in the context of two important cybersecurity scenarios, namely vulnerability analysis and threat modeling, using a standard set of performance measures. The evaluation benchmark of our proposed cybersecurity analysis approach involves an extensive database that comprises over 130 000 samples stemming from 8 actual online cybersecurity repositories and other corresponding manufacturer websites, which is the result of *Research Process 2*.

Table 4.5: Research process 3: bridging the data gap

Adopted Methods	Deliverables	Supporting Knowledge
Step 1: industrial surveys.	Initial Problem: human-centred and subjective vulnerability analysis lead to errors and inconsistencies.	Vulnerability-driven cybersecurity.
Step 2: literature review.	Research Question: can the manual procedures of severity assessment and classification of security vulnerabilities be supported by ML algorithms?	Cybersecurity domain knowledge; AI methods for cybersecurity; Quality of database.
Step 3: contemporary practices review.	Requirements Specification: identifies vulnerability patterns, from multivariate time-varying data.	Existing ML methods for cybersecurity; Cybersecurity domain knowledge; Online cybersecurity data-sources.
Step 4: ML-based multi-dimensional vulnerability analysis.	Initial <i>Artifact II-A</i> : LR-based vulnerability classifiers for each CVSS metric. Initial <i>Artifact II-B</i> : LSTM-based weakness categorization using CWE abstracts.	Online cybersecurity data-sources; Text-mining techniques; Natural language processing techniques.
Step 5: <i>Study VI &amp; VIII</i> .	Artifact Evaluation: performance trade-offs compared to similar works.	ML-model evaluation metrics.
Step 6: <i>Study VII</i> .	Artifact Evaluation: CI vulnerability assessment in terms of vulnerability severity.	Cybersecurity domain knowledge; ML methods for cybersecurity.
Step 7: updated question.	Updated Problem: no one-size-fits-all solution that matches multiple vulnerability assessment requirements.	Cybersecurity domain knowledge; ML methods for cybersecurity.
Step 8: literature review.	Updated Question: how to find an optimized model from a model-set for a given cybersecurity task?	Cybersecurity domain knowledge; ML model evaluation & optimization; Quality of database.
Step 9: contemporary practices review.	Updated Requirements: streamline vulnerability assessment process and further improve <i>Artifact I, II-A, and II-B</i> .	Existing ML methods for cybersecurity; Cybersecurity domain knowledge; Vulnerability-analysis lifecycle; Online cybersecurity data-sources.
Step 10: a selective ensemble approach.	<i>Artifact II-C</i> : an ensemble approach that streamlines the process of vulnerability-assessment model training, evaluation and optimization.	Online cybersecurity data-sources; Text-mining techniques; Ensemble models.
Step 11: a pipeline of ML models.	Implemented Artifact: selects the most effective ensemble models from five base machine-learning algorithms for two different cybersecurity tasks.	Text-mining techniques; ML-model evaluation metrics; Ensemble models.
Step 12: <i>Study IX</i> .	Artifact Evaluation: performance trade-offs compared to similar works.	ML-model evaluation metrics; Ensemble models.
Step 13: publication.	Artifact knowledge: communicated through Papers III, V and VII.	Cybersecurity research community and stakeholders.

The experimental results show improved prediction performance compared to individual text-mining techniques, commonly employed in security management when analyzing vulnerability reports. For example, LSTM-RNN (Long Short-Term Memory based Recurrent Neural Network) (Zhou et al., 2016) and MLP (Multi-layer Perception Classifier) (Zanaty, 2012) are chosen as base learners for threat classification ensemble, while LSTM-RNN, NBSVM (Naïve Bayes (Wang and Manning, 2012) based Support Vector Machines) (Joachims, 2001) and MLP are chosen as base learners for vulnerability-severity score computation ensemble.

#### 4.2.4 RESEARCH PROCESS 4: VULNERABILITY ANALYSIS FOR CRITICAL INFRASTRUCTURES

We conducted a literature review on IT and OT semantics for CI cybersecurity protection, and summarized the results in sub-section 3.3.1. We discovered that there is a lack of taxonomy that is concise, easy to use, and free of overlapping concepts and features. This is partly because IT security and OT security utilize distinct terminologies. There is also limited support for query-based dependence analysis in previous CI models. Therefore, we set up the goal of modeling IT/OT convergent CI semantics that is extendable in terms of component types and their interconnections, while facilitating scalable dependence assessment.

This research process aims to answer the question of how to model CIs that support vulnerability analysis, and further, how to assess CI vulnerabilities with the support of the previous artifacts (i.e., the correlated vulnerability database and the ensemble ML pipeline). To solve this question, we proposed an extensible taxonomy for CPS-based CIs that supports static analysis of functionally dependent components utilizing deductive rules, as well as vulnerability chain modeling and identification. Steps of this research process are summarized in Table 4.6.

We define the characteristics of IT and OT entities and their convergence across the cyber-physical layers of CIs, as CI cybersecurity is a relatively immature domain. Our taxonomy is built upon the semantic models and industrial frameworks introduced in Section 3.3.1, following the Telos (Mylopoulos et al., 1990) (Koubarakis et al., 2021) language, while also inspired by the architecture analysis and design language (AADL) (Feiler, Lewis, and Vestal, 2003). We define our dependence rules on top of the works introduced in Section 3.3.3. Similarly, instantiations of the proposed taxonomy are built on top of existing smart grid models introduced in Section 3.3.2, which results in multiple reference models to enhance the extendability and explanatory strength (Nickerson, Varshney, and Muntermann, 2013).

By carrying out this procedure, we can obtain a varied collection of attributes and dimensions of CI items concerning cybersecurity. Our taxonomy is visualized using the open-source tool ConceptBase (Jarke et al., 1995). This tool enables the representation of classes, domain-specific objects, and instantiated models within the same database. Additionally, it supports the specification of graphical symbols for particular classes, which are then applied to all instances of that classes. Moreover, the defined models can be simply extracted in a preferred format (e.g., XML) and used for power-grid simulation and vulnerability modeling.

We further proposed *Artifact IV* to enhance the level of automation in system-wide vulnerability assessment.

We validated the applicability and expressiveness of our artifacts in different CI scenarios in power grids and manufacturing systems, including *Study X* and *Study XI*.

Table 4.6: Research process 4: vulnerability analysis for critical infrastructures

Adopted Methods	Deliverables	Supporting Knowledge
Step 1: literature review.	Initial Problem: lack of standardized CI taxonomy for cybersecurity analysis.	CI cybersecurity domain.
Step 2: literature review.	Research Question: how to model CI networks to support vulnerability analysis?	CI domain knowledge; Model-based cybersecurity analysis.
Step 3: contemporary practices review.	Requirements Specification: CI components are modeled using meaningful concepts and relationships, to support the analysis of complex cyber-physical correlations.	Existing CI taxonomies; CI conceptual models; CI dependence analysis; Reference models for CIs protection.
Step 4: CI ontological modeling.	Initial <i>Artifact III</i> : specification repository for CI models and internal CI dependencies.	Conceptual modeling; Inventory ontology; Model-based system engineering.
Step 5: CI information collection and modeling.	Implemented Artifact: a repository of power-grid and manufacturing networks for static vulnerability analysis.	Power-grid domain knowledge; Manufacturing industry knowledge; Model-based system engineering.
Step 6: <i>Study X</i> in power grids.	Artifact Evaluation: structural, functional adequacy, compatibility, operability, reliability, and maintainability of CI dependence analysis and cascade modeling.	Power-grid domain knowledge; Model-based system engineering.
Step 7: updated question.	Updated Problem: system-wide vulnerability assessment needs enhanced levels of automation.	Cybersecurity domain knowledge; ML methods for cybersecurity.
Step 8: literature review.	Updated Question: how can vulnerabilities of complex CIs be modeled and assessed with the support of a curated database and vulnerability assessment algorithms?	Cybersecurity domain knowledge; ML methods for cybersecurity; Model-based system engineering.
Step 9: contemporary practices review.	Updated Requirements: vulnerability analysis orchestration to connect <i>Artifact I, II, and III</i>	Cybersecurity domain knowledge; Vulnerability-analysis lifecycle; Model-based system engineering.
Step 10: streamlining vulnerability analysis.	<i>Artifact IV</i> : method for streamlining vulnerability analysis for CIs.	Model-based system engineering.
Step 11: <i>Study XI</i> in manufacturing.	Artifact Evaluation: applicability for CI cybersecurity in a query-able taxonomy.	Manufacturing industry knowledge; CI cybersecurity.
Step 12: dissemination and presentation.	Artifact knowledge: Paper I, Paper VI and Paper VIII.	CI research community; Power-grid stakeholders; Manufacturing stakeholders.

*Study X* evaluates structural, functional adequacy, compatibility, operability, reliability, and maintainability of the taxonomy. More details of these metrics are presented in subsection 4.3.5 .In the context of these six metrics, the expressiveness of this taxonomy is evaluated using a standard IEEE 9-bus model. Explicit information such as heterogeneous components of the IEEE 9-bus model and connections between these components are defined as instantiations of the CI taxonomy. In addition, we present deductive rules

and queries to validate certain semantic relations, such as data and power connections.

*Study XI* evaluates further the applicability and usefulness of the proposed taxonomy in terms of human-in-the-loop vulnerability assessment. In addition, we implemented a human-robot collaboration (HRC) assembly model based on a real-world system located at the ASSAR Industrial Innovation Arena in Sweden. We interviewed industrial production personnel to get information regarding the general manufacturing network structure and this specific HRC assembly method. This stage ensures that the topological factors and other settings of our instantiated manufacturing models correspond to a realistic scenario of industrial manufacturing processes. Simultaneously, historical instances of vulnerability are analyzed to derive statistical patterns and severity CVSS scores, which are then used to construct attack graphs using simulation tools.

## 4.3 DISCUSSIONS ON TRUSTWORTHINESS OF THE RESEARCH

The US National Security Agency technical director emeritus Robert Meushaw (2012) wrote in a forum about what distinguishes security-science research and practice is that the former contributes to knowledge gain of the community. He states that security-science research should be based on objective and qualitative or quantifiable descriptions of security properties and behaviors. Such descriptions should clarify the limitations clearly. The US Department of Defense also emphasizes in their JASON report *Science of Cyber-Security* Corporation (2010) the importance of shifting security practices towards the science of security. According to this JASON assessment, foundational concepts, mathematical constructions, or techniques for accurate prediction and measurement are necessary to strengthen the current scientific foundation of cybersecurity.

This thesis utilizes mixed research methods. And hence, the trustworthiness validation of this thesis considers suggestions in both qualitative (Thomson, 2011) and quantitative (Heale and Twycross, 2015) research evaluation metrics. Considering these suggestions, the thesis outcomes are evaluated from five main perspectives, namely (i) repeatability and reproducibility; (ii) validity and generalization; (iii) applicability and usability; (iv) ethical concern; and (v) ontology evaluation metrics.

### 4.3.1 REPEATABILITY AND REPRODUCIBILITY

Maxion (2011) underlines the significance of experiments in the cybersecurity field, particularly its evaluation criteria, such as repeatability and reproducibility. Repeatability and reproducibility metrics have also been stressed by Collberg and Proebsting (2016). Repeatability implies that the experiment should produce consistent and progressing results, if repeated. Reproducibility suggests that by following the experiment steps introduced, researchers can reproduce the same results. Validity describes the proposed theory/model should be well-grounded and generalizable.

Detailed instructions are given to record the experiment steps and case-study procedures in this thesis, to increase repeatability and reproducibility. For example, step-by-step documentations are provided to guide the implementation and use the proposed correlated and cross-linked vulnerability database proposed in Chapter 6. Some correlated vulnerability files are also uploaded to the GitHub platform as Jiang (2021) project for open source usage and to enhance reproducibility (Stodden, 2010) (González-Barahona and Robles, 2012). Data pre-processing and ML model training processes are also clearly

documented in Chapter 7, with several trained models uploaded to Github as *Vulnerability Classifier* (2021) for academic purposes.

### 4.3.2 VALIDITY AND GENERALIZATION

Herley and Van Oorschot (2017) describe that a claim should be consistent with other claims or available observations. In addition, the predictions of proposed models should be correct, so that the number of observations that can be accurately anticipated should increase and advance through time. Limitations in the earlier scientific security research efforts include a lack of formal verification methods and an excessive emphasis on symbolic interpretation instead of requirement analysis of real implementations. Meanwhile, they discuss some future directions for security science, such as formal mathematical models and language, data collection, good experimental design, and quantitative and qualitative metrics and measurements that deserve more consideration in security research.

Explicit definitions of concepts are provided in this thesis to enable common understandings to further enhance validity and generalization. Such definitions are based on relevant literature reviews and meetings with field stakeholders.

The data gathered to design the artifacts of this research is based on different sources such as observations, interviews, and the study of organizational documents related to security-analysis processes, as well as existing scientific literature. For example, the data gathered for vulnerability assessment is collected from multiple standard data sources. The datasets used for ML-artifacts evaluation and demonstration are randomly sampled from the available historical vulnerability instances. Moreover, datasets used for ML-model training and testing are duly clarified for the sample size and how the samples are generated. Based on a thorough review of the literature, a set of evaluation metrics such as accuracy and F1-score are carefully chosen for each ML classification task.

Cybersecurity knowledge needs to be appropriately generalized to allow sharing on such a fundamental basis. Spring, Moore, and Pym (2017) put attention on how to draw useful generalizations in scientific security. They state that structured observations performed with robust research methods such as randomized controlled trials can overcome the potential lack of control groups in experimental studies. They also acknowledge that case studies and model-based reasoning are alternative approaches to be used in the cybersecurity area. The proposed artifacts in this thesis include both models and instantiations to allow generalization such as the application of the proposed models to other related fields. For example, the proposed CI taxonomy (or *Artifact III*) describes the basic CI components and their dependencies. Instantiations of this taxonomy not only cover the introduced scenarios about power grids and manufacturing processes, but can also extend to other technical critical infrastructures.

### 4.3.3 USEFULNESS AND USABILITY

One research gap in security science pointed out by Corporation (2010) is that translating scientific developments into practice is missing. This gap has been highlighted by Degabriele, Paterson, and Watson (2010) and Maughan et al. (2013), as a system that is proved to be secure in theory may not be fully trustworthy in real environments. Vulnerability-centered cybersecurity research must adapt to real-world circumstances, integrate with user workflows, and provide practical benefits (Heelan, 2011). As cybersecurity is tightly tied to human behavior and the examined systems, the evaluation of cy-

bersecurity artifacts is typically domain-specific and context-specific (Evans and Stolfo, 2011).

This thesis explores various views on the utility of the proposed artifacts. For example, the utility of the instantiated cross-link vulnerability repository is assessed based on the data sources, accessible features, structured scheme, integrated standards, and query assistance provided for vulnerability analysis.

Feedback is solicited from security professionals and also operational practitioners in power grids and manufacturing, to enhance the usefulness of this thesis. For example, *Study IV* is carried out locally to evaluate applications of the correlated vulnerability database in assessing historical vulnerability reports, to further raise applicability prospects. Similarly, *Study XI* assesses the applicability of the proposed CI taxonomy and cybersecurity orchestration framework in an actual human-robot collaboration system.

This thesis also considers the usability that is defined as “*the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*” by Bevan, Carter, and Harker (2015). Following this definition and other usability properties described in Bevan et al. (2016), this thesis brings primary benefits to security actors involved at various SOC levels. Several beneficiary use cases are discussed in the following empowerment.

- Security operators can utilize the proposed common data model and correlation techniques to implement their localized database, to get up-to-date vulnerability records for further analysis. By doing so, security operators get support for the development of vulnerability patches and deploy them before exploits are made available.
- Security analyzers can get support from the proposed ML-based vulnerability analysis method in vulnerability categorization, assessment and remediation. For example, vulnerability-metric measurements are collected and computed automatically without conventional manual inputs, which alleviates these security actors’ tasks.
- Operators of CI networks, especially power-grid and manufacturing networks, can apply the proposed CI taxonomy and instantiated model to enhance the system-structure design. They may develop further cybersecurity simulations to identify vulnerabilities and analyze related impact scenarios that may bring damage to critical CI assets once exploited.
- SOC organizations can benefit from the proposed cybersecurity orchestration solution to achieve a higher level of automation in vulnerability-severity awareness and risk-analysis exercises, and also to enhance cybersecurity situation-awareness between different organizational roles.

#### 4.3.4 ETHICAL CONCERN

Dittrich, Kenneally, et al. (2012) provide the *Menlo Report* to address some ethical constraints in information and communication technology researches, including cybersecurity works. They restate the three classic concerns brought up in the *Belmont Report* to respect for persons, beneficence, and justice. This first principle allows applications of ethical principles to protect humans from activities having human-harming potential, even though those activities may not directly involve humans. The second principle implies the need for systematic assessments of both risks of harm and benefit. The third

principle suggests that the selection and treatment of both participating and impacted subjects should be fair. Dittrich, Kenneally, et al. (2012) also clearly clarify one more consideration: respect for law and public interest. This fourth principle addresses the concern to legal controls of confidentiality, availability, and integrity involving information or information systems.

Following these four ethical principles, the research subjects are carefully considered in light of the above principles. The data collected is mainly from online open-source repositories and is stored and managed with relatively high authorization requirements. The research outputs regarding details of the companies are not published considering security and privacy properties.

#### 4.3.5 ONTOLOGY EVALUATION METRICS

We validate the structural, functional adequacy, compatibility, operability, reliability and maintainability of our taxonomy in two studies of instantiated power grid models, following standards of Nickerson, Varshney, and Muntermann (2013) as well as the ontology quality evaluation and requirements (OQuaRE) (Duque-Ramos et al., 2014) framework. The OQuaRE framework adopts the ISO/IEC standards for software product quality requirements and evaluation proposed by Suryan, Abran, and April (2003) in ontology assessment.

- The structural metric measures semantic models from four dimensions: (i) whether an ontology has a high cohesion with strongly related classes and a good domain coverage; (ii) whether an ontology is informative; (iii) whether an ontology provides formal relations support; and (iv) whether an ontology is related to the existence of multiple inheritances.
- The functional adequacy metric expects an ontology to have characteristics as: (i) avoiding heterogeneous terms; (ii) providing consistent search and query; (iii) representing acquired knowledge clearly; and (iv) can be used to build other ontology.
- The compatibility metric considers the performance of an ontology when adapted to different environments without additional actions other than those that were clarified by the ontology (i.e., adaptability).
- The operability metric assesses the effort and knowledge required for individual users to make use of the ontology.
- The reliability metric evaluates the performance of an ontology under specified conditions for a given period of time, which is divided into availability and recoverability.
- The maintainability metric measures how well an ontology can adapt to changes in the environment, such as changes in requirements or functional specifications. This metric has multiple dimensions, one is reusability which estimates the degree an ontology can be used for building other assets, and the other is modularity which tests the impact on an ontology when changes to partial components are made.



## CHALLENGE IDENTIFICATION AND BASELINE STUDIES



# CHAPTER 5

# CHALLENGE IDENTIFICATION AND BASELINE STUDIES

Vulnerability-related information is usually reported in natural-language expressions across numerous and multifaceted repositories, as well as hundreds of manufacturing websites, and other thousands of security blogs posted every day (Sauerwein et al., 2019). Multiple cybersecurity reports and bulletins provide various perspectives for vulnerability analysis. For instance, data sources such as *CVE*, *CERT* and *Shodan* are commonly adopted in academic research and industrial products. However, these information items are separated, stored and published in different data formats. Furthermore, they obey different standards, each with its own syntax and semantics.

This chapter addresses the challenges in CI vulnerability assessment using publicly accessible cybersecurity repositories. This research question can be divided into three inter-related sub-questions as follows: (i) how to access relevant cyber security data sources? (ii) how to interpret the corresponding data features? and (iii) what are the challenges in deploying these repositories to support CI cyber security from a user's perspective?

To answer the first and the second sub-questions, we carried out a baseline study (*Study I*) across multiple open and public cybersecurity repositories to explore the syntax and characteristics of these data sources. This baseline study explores cybersecurity data sources from a general user's perspective. Subsequently, we conducted an industrial survey (*Study II*) to discuss CI cybersecurity stakeholder perspectives on the usage of open and public cybersecurity repositories and instrumenting CVSS. This survey attempts to answer the third sub-question, together with a vulnerability analysis case study (referred to as *Study III*) on an actual CI system in order to assess the quality of available vulnerability data collected from open and public cybersecurity repositories. In doing so, we emphasize particularly the incompleteness and inconsistency of *CVE*, *NVD* and vendor-specific security repositories. We also interviewed several security specialists from the investigated institution to collect their feedback on the case study results and related challenges in CI vulnerability assessment.

This chapter covers, but is not limited to results from *Paper IV*.

## 5.1 BEFORE THE BASELINE STUDY

The first crucial step is to decide the data relevance, or the level of consistency between the data content and the user's area of interest.

### 5.1.1 DATA IDENTIFICATION

This thesis focuses on vulnerability-driven cybersecurity assessment of complex CIs. Cybersecurity activities around vulnerabilities can be divided into different stages of the

vulnerability lifecycle, as introduced in Section 2.3. Based on the introduced vulnerability lifecycle (Joh and Malaiya, 2011) (Frei et al., 2006), vulnerability-driven cybersecurity follows a continuum from vulnerability identification and vulnerability categorization to vulnerability assessment and vulnerability remediation, and then loops back to the beginning step to track another vulnerability. This closed-loop model illustrates the feedback from vulnerability validation that prompts a patching action and/or tracking another vulnerability. Information related to each step of this closed-loop needs to be collected for security patterns analytic.

Early detection or identification of vulnerabilities minimizes the *window of exposure* (Frei et al., 2006), which enhances the security of a system. Collected system information is synchronously compared with a vulnerability-signature database to identify vulnerabilities. Such vulnerability signatures refer to official instances disclosed in *CVE* and *NVD*, and unofficial security reports in news, forums and research publications. Security news posted in Twitter (Sabottke, Suciu, and Dumitras, 2015), for example, are usually up-to-date, but might not be reliable. Additionally, system-relevant information is retrieved from *CCE* and *CPE* to support the vulnerability identification process.

The vulnerability-categorization approach requires the exploration of reported vulnerabilities in a vulnerability database to identify vulnerability patterns. This thesis utilizes *CWE* to derive a comprehensive weakness classification taxonomy of known vulnerabilities.

Vulnerability assessment plays a key role in patching prioritization and decision making. It is further categorized into qualitative methods and quantitative methods. The qualitative method primarily addresses relationships between vulnerability and risk. It expresses vulnerability observations based on non-numerical data, which is supported by threat categorization extracted from the website *cvedetails.com* and attack information is provided by *CAPEC* and *ATT&CK*. The quantitative assessment, on the other hand, measures the likelihood and impact of a vulnerability to prioritize patching exercises (Allodi and Massacci, 2017), which is empowered with *CVSS* severity scores retrieved from *NVD* and some third party analysis.

Vulnerability remediation may involve various strategies, such as updates, patches, and improving system access control. The remediation level of a given vulnerability instance is assessed based on whether workarounds, temporary fixes, or official patches are available. The remediation availability information is usually collected through vendor websites. Meanwhile, *CRE* (although not in use anymore) also provides some general guidelines for remediation.

Vulnerability validation is conducted through simulation analysis (Negi et al., 2019) or interviews with cybersecurity experts. This thesis utilizes vendor websites, some third-party analysts and security research publications as sources for vulnerability validation information.

The aforementioned data sources are further categorized them into object classes to represent these keywords and their correlations, in order to recognize security patterns between different directions such as *instance*, *weakness*, *threat*, *attack*, *remediation*, and *system*. These objects are also mapped to the data sources in Table 5.1. We would discuss these data sources with more details next.

Table 5.1: Summary of vulnerability data sources

<b>Data Category</b>	<b>Object</b>	<b>Purpose</b>	<b>Data Source</b>	<b>Description</b>
Standard	System	Vulnerability Identification	CCE; CPE	Contains product records of operating systems, software and hardware.
Standard	Threat	Vulnerability Assessment	CVEDetails	Contains common threat types like malware and web-targeted threats.
Standard	Weakness	Vulnerability Categorization	CWE	Contains commonly occurring weaknesses in software.
Standard	Attack	Vulnerability Assessment	CAPPIC; ATT&CK	Contains attack pattern specifications, e.g., techniques and procedures.
Standard	Remediation	Vulnerability Remediation	CRE	Contains recommended countermeasures.
Instance	Vulnerability	Vulnerability Identification Vulnerability Validation Vulnerability Assessment	CVE; NVD; CERT-VND	Contains officially disclosed vulnerability reports; NVD also provides CVSS severity scores.
Instance	Mixed	Vulnerability Identification Vulnerability Remediation Vulnerability Validation	Vendor Website	Contains official vulnerability disclosure related to this manufacturing.
Instance	Mixed	Vulnerability Identification	Security News	Contains unofficial security issues, usually up-to-date.
Instance	Mixed	Vulnerability Identification	Security Forum	Contains both vulnerability reports and unofficial security discussions.
Instance	Mixed	Vulnerability Identification Vulnerability Validation	Security Research	Contains reliable security research, but might not up-to-date.

### 5.1.2 DATA COLLECTION

This thesis employs multiple ways to collect vulnerability data from online public repositories, as depicted in Figure 5.1. The collected datasets are stored separately with the corresponding labels. Figure 5.1 also illustrates the format of stored datasets as one of the markup language tags (e.g., XML, HTML), or JSON format, or CSV format. For instance, CVE is available in CSV, HTML, Text, XML, JSON and CVRF, among which we selected JSON for baseline studies of CVE.

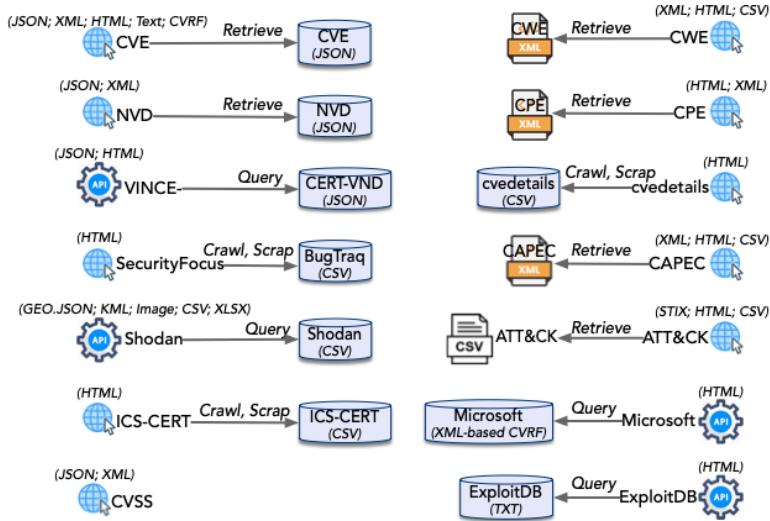


Figure 5.1: Example of vulnerability data collection

More specifically, CVE data is downloaded in JSON format. CVE assigns an ID to a vulnerability instance, together with a description of this instance and some references linked to the vendor or some third-party analysts. The local CVE database is updated on an hourly schedule to synchronize with the online CVE data feeds.

NVD data feeds are directly downloaded and stored in a local database in JSON format. The JSON format is an open standard file format used for interchanging data, consisting of human-readable text (i.e., not binary) attributes' value pairs. JSON objects can be nested inside other JSON objects. In contrast, each nested object has a unique access path across the tree-like structure. The proposed curator also sets up a scheduler to perform hourly data retrieval and update through an existing Python library *APScheduler* (referring to Advanced Python Scheduler), to ensure that the local files and online NVD data feeds are synchronized. An hourly schedule is chosen to mirror NVD data considering that the “*recent*” and “*modified*” feeds in NVD are updated every two hours, while the rest are updated nightly.

Standard enumerations datasets like CWE and CAPEC lists are directly downloaded from the websites in XML format, with CWE in version 4.6 and CAPEC in version 3.6 (latest check date is November 3, 2021). CWE and CAPEC apply similar hierarchical structures where the top categories contain tree-structure patterns, as introduced earlier in Section 2.2.

We utilized web crawling and web scraping techniques Mahto and Singh, 2016 to capture vulnerability information published in security analysis’ websites like SecurityFo-

cus. Web crawling refers to the process of browsing and indexing contents from web pages. Examples of relevant built-in Python functions include *urllib.request* that downloads *html* pages and *urllib.error* that handles exceptions. Web scraping means locating and collecting certain information using tools like HTML parser *Beautiful Soup* (2022). Information retrieved from fetching, parsing and extracting targeted data is stored in a proper format tailored to the targtted information usage. For instance, the information extracted from SecurityFocus is stored in local files with fields like Bugtraq-ID, CVE-ID, title, publish date and affected product in CSV (or comma-separated values) format.

Some manufacturers maintain their security repositories and provide APIs towards these repositories. For instance, *Microsoft* security updates are accessible through an official *MSRC Security Updates API* (2022), and can be further retrieved in CVRF format documents. A *Microsoft* CVRF document may be composed of several vulnerability containers, each of which refers to one *CVE* instance. Additionally, there are open-source Github projects (e.g., *VIA4CVE - Vulnerability Information Aggregator for CVEs* (2022)) that query this *Microsoft* API and fetch *Microsoft* bulletins and vulnerability data in JSON format. There are also third-party analyzers such as *Shodan* that provides API to query CI relevant vulnerabilities using an academic-level *Shodan* account.

## 5.2 STUDY I: DIVERSITY AND HETEROGENEITY OF VULNERABILITY DATA

We conducted some static analysis of retrieved *NVD* data feeds in JSON format from the *NVD* website at 15:00 pm on 2021 November 3rd. We uploaded some of the marked up data feeds and adopted analysis methods as a Github project in Jiang (2021). The retrieved 20 JSON files (with file name '*nvdCVE-1.1-year.json*' with year value in the range from 2002 to 2021) contribute to 173 365 vulnerability instances. This vulnerability set contains 9821 vulnerability instances marked as “Rejected”, meaning that one specific vulnerability instance is not accepted as a *CVE* entry with reasons like duplicated entry or withdrawal by the original requester. After removing the “Rejected” entries, the data set contains 163 505 instances. Next we discuss the static patterns of disclosed vulnerabilities in *NVD* with some charts generated in this study.

Figure 5.2 visualizes the vulnerabilities published in *NVD* in a daily manner. The amount of published vulnerabilities has been increasing, particularly in the past five years. The daily published amounts peaked at 1098 instances on December 31, 2004, followed by 816 and 791 instances published on May 2, 2005 and December 31, 2002, separately.

Till November 3, 2021, *NVD* assigns 163 375 vulnerabilities with *CVSS* V2 base scores together with specific *CVSS* V2 vectors. Meanwhile, *NVD* assigns 90 008 vulnerabilities (published after 2015) with *CVSS* V3 base scores together with specific *CVSS* V3 vectors, to support assessment of the exploitability and impact of published vulnerabilities. Figure 5.3 and Figure 5.4 present the distribution of *CVSS* scores and *CVSS* metrics along the published years under *CVSS* V2 and *CVSS* V3, separately. A small fragment of vulnerability instances published in 2021 have no *CVSS* V2 scores or vectors. Similarly, some vulnerabilities published in 2016 and 2021 are not assigned with *CVSS* V3 scores or vectors.

Considering *CVSS* qualitative scales, a large portion of vulnerabilities published in *NVD* are within the ‘medium’ V2 severity scale, as illustrate in Part (a) of Fig. 5.3, which is converted to similar slice-sizes of ‘medium’ and ‘high’ V3 severity scales shown in Part (a) of Figure 5.4.

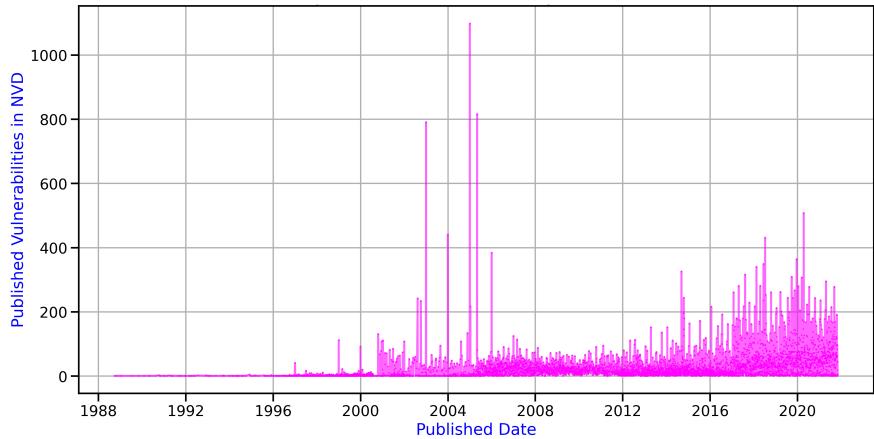


Figure 5.2: Daily published vulnerabilities on NVD from 1999 to November 3, 2021

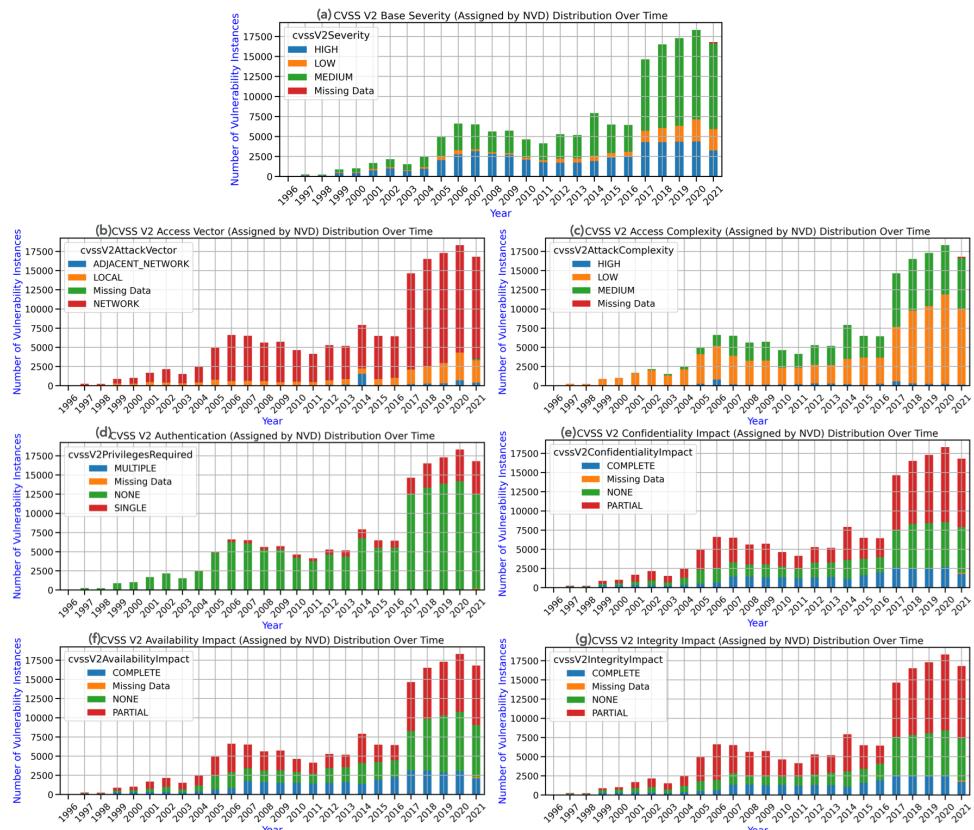


Figure 5.3: CVSS version 2 score and metric value distribution for NVD entries

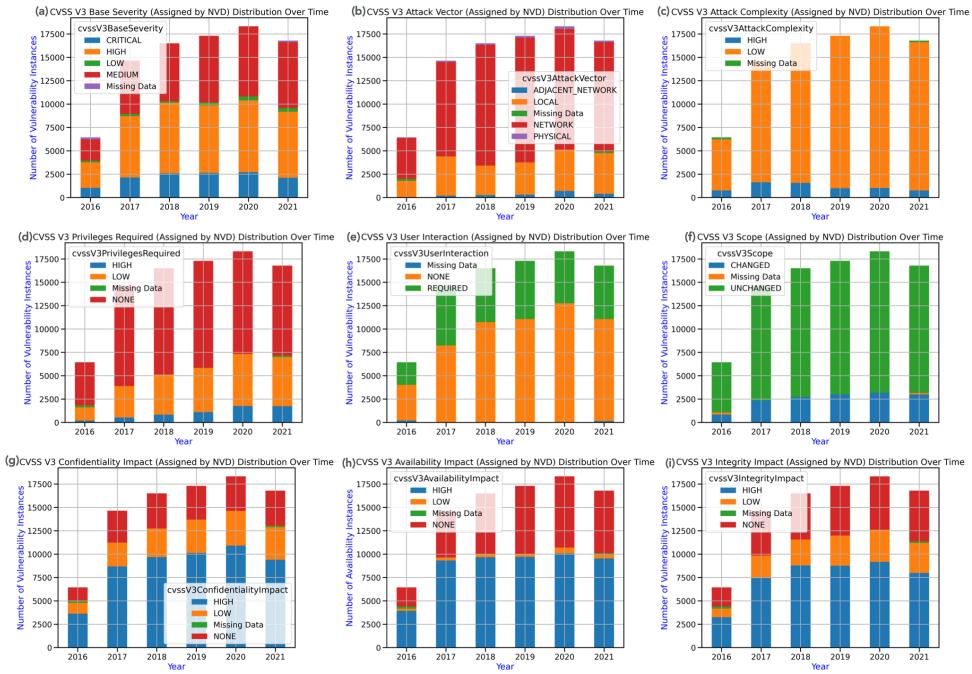


Figure 5.4: CVSS version 3 score and metric value distribution for NVD entries

We discuss the vulnerability patterns when only taking the NVD assigned CVSS base properties into consideration. Note that CVSS base properties do not consider the vulnerability characteristics that may change over time or in different deployment environments. And hence, the discussions here do not target specific systems. As shown in the exploitability distribution under CVSS V2 metrics in Part (b), (c), and (d) of Figure 5.3, most of the reported vulnerability instances are highly exploitable and do not require extra conditions to exploit. The majority of vulnerability instances are accessed through a network and are therefore remotely exploitable, while the minority of vulnerability instances require local access or a local account. Vulnerability instances that require adjacent network access have been reported only since 2012. Such vulnerabilities appear in a comparably high frequency in the year of 2014. Generally, the knowledge level and skills to trigger a successful attack are increasing, as evidenced by the trend of a more even distribution between low complexity and medium complexity over the past ten years. Meanwhile, attackers need to authenticate none or only one time to exploit most of the existing vulnerability instances. The remaining parts, i.e., Part (e), (f) and (g) of Figure 5.3 illustrate the distribution of impact severities of the retrieved vulnerability instances under CVSS V2, which indicates a higher diversity of impact compared to the diversity of exploitability in general. The distributions of confidentiality impact, integrity impact and availability impact also show similar diversity patterns, in which *Partial-* and *None-*impact have more appearances than *High-*impact.

We found similar exploitability and impact patterns in vulnerability distributions under CVSS V3 metrics, as shown in Figure 5.4. Vulnerabilities exploited by physical-path based attacks have limited presence throughout the time range between 2016 to 2021. Exploiting most of these published vulnerabilities does not require privileges or user interaction.

NVD maps CVE vulnerabilities to a selected set of 241 unique CWEs (introduced earlier in sub-section 2.2.2) in *NVD CWE Slice* (2021). Similar to the statistic analysis for CVSS mapping, we utilize tools like *Pandas* and *Numpy* to analyze the static patterns of CWE mappings in NVD vulnerabilities. We found that in the set of 163 505 vulnerabilities (with “Rejected” entries removed), 44 970 vulnerabilities have no CWE related information that is further consists of 3 scenarios where (i) 26 739 vulnerability instances are assigned with *NVD-CWE-Other* only, e.g., *CVE-2021-0309*; (ii) a vulnerability instance is assigned with *NVD-CWE-noinfo* only, e.g., *CVE-2021-0377*; and (iii) CWE entries of 161 vulnerability instances are missing, e.g., *CVE-2021-1117* that has no CWE weakness information till the statistic analysis conduction time in November 3, 2021. The remaining 118 535 vulnerability instances with some CWE information contains 115 469 instances assigned with single CWE entries, 2 934 instances assigned with 2 CWE entries, 116 instances assigned with 3 CWE entries, and 4 instances with 5 CWE entries. For instance, the vulnerability *CVE-2020-11901* (with report “*The Treck TCP/IP stack before 6.0.1.66 allows Remote Code execution via a single invalid DNS response.*”) is assigned 4 different CWE labels, namely *CWE-330* (i.e., use of insufficiently random values), *CWE-787* (i.e., out-of-bounds write), *CWE-125* (i.e., out-of-bounds read) and *CWE-131* (i.e., incorrect calculation of buffer size). Some of examples are listed in Table 5.2 to show various ways of mapping from vulnerability instances to CWE and further to CAPEC. Further, the 10 weakness types with top occurrences in NVD published vulnerabilities are *CWE-79* assigned to 16 903 instances, *CWE-119* assigned to 11 197 instances, *CWE-20* assigned to 7 948 instances, *CWE-89* assigned to 6 787 instances, *CWE-200* assigned to 6 421 instances, *CWE-264* assigned to 5 128 instances, *CWE-787* assigned to 4 318 instances, *CWE-22* assigned to 3 931 instances, *CWE-125* assigned to 3 418 instances, and *CWE-352* assigned to 3 158 instances.

Table 5.2: Examples of mappings from vulnerabilities to CWE and CAPEC

CVE-ID	CWE-ID(s)	CAPEC-ID(s)
<i>CVE-2018-8792</i>	<i>CWE-125</i>	<i>CAPEC-540</i>
<i>CVE-2021-27438</i>	<i>CWE-94</i>	<i>CAPEC-242, CAPEC-35, CAPEC-77</i>
	<i>CWE-798</i>	<i>CAPEC-191, CAPEC-70</i>
<i>CVE-2016-4309</i>	<i>CWE-362</i>	<i>CAPEC-26, CAPEC-29</i>
	<i>NVD-CWE-Other</i>	N/A
<i>CVE-2018-20314</i>	<i>CWE-362</i>	<i>CAPEC-26, CAPEC-29</i>
	<i>CWE-125</i>	<i>CAPEC-540</i>
	<i>CWE-787</i>	N/A
<i>CVE-2017-8535</i>	<i>CWE-119</i>	<i>CAPEC-10, CAPEC-100, CAPEC-123, CAPEC-14, CAPEC-24, CAPEC-8, CAPEC-42, CAPEC-44, CAPEC-45, CAPEC-46, CAPEC-47, CAPEC-9</i>
	<i>CWE-369</i>	N/A
	<i>CWE-476</i>	N/A
	<i>CWE-674</i>	<i>CAPEC-230, CAPEC-231</i>
<i>CVE-2019-10084</i>	<i>CWE-311</i>	<i>CAPEC-157, CAPEC-158, CAPEC-204, CAPEC-31, CAPEC-37, CAPEC-383, CAPEC-384, CAPEC-385, CAPEC-386, CAPEC-387, CAPEC-388, CAPEC-477, CAPEC-609, CAPEC-65</i>
	<i>CWE-330</i>	<i>CAPEC-112, CAPEC-485, CAPEC-59</i>
	<i>CWE-732</i>	<i>CAPEC-1, CAPEC-122, CAPEC-127, CAPEC-17, CAPEC-180, CAPEC-206, CAPEC-234, CAPEC-60, CAPEC-61, CAPEC-62, CAPEC-642</i>
	<i>CWE-532</i>	<i>CAPEC-215</i>
	<i>CWE-384</i>	<i>CAPEC-196, CAPEC-21, CAPEC-31, CAPEC-39, CAPEC-59, CAPEC-60, CAPEC-61</i>

Historical vulnerabilities disclosed in NVD exemplified the diversity of assigned weak-

nesses. For example, the *Category* weakness of “*Permissions, Privileges, and Access Controls*” (*CWE-264*) is assigned to 5128 vulnerability instances, as shown in Table 5.3. *CWE-264* is eliminated from the Research Concept View (*CWE-1000*). Instead, this view (*CWE-1000*) distributes weakness related to features of improper permission management and improper privilege management to a list of *CWE-ID(s)*. For instance, *CWE-284* (or improper access control) is one of the most abstract *Pillar*-level weaknesses in *CWE-1000*. *CWE-284* is allocated to 990 vulnerability instances. Further, another view, or *CWE-700* and *Seven Pernicious Kingdoms* adopt a *Category* weakness *CWE-254* that covers *CWE-284*. The description of *CWE-254* is “*Security Features*” that is any feature like improper authentication or cryptography issue. Even as vague as *CWE-254*, this weakness is assigned to 419 vulnerability instances disclosed in *NVD*.

Table 5.3: Example of diverse *CWE-ID*s allocated to vulnerability instances

<i>CWE-ID</i>	Abstraction Level	Name	Occurrence	Membership
<i>CWE-264</i>	Category	Permissions, Privileges, and Access Controls	5128	<i>MemberOf CWE-635</i>
<i>CWE-254</i>	Category	7PK - Security Features	419	<i>MemberOf CWE-700</i>
<i>CWE-284</i>	Pillar	Improper Access Control	990	<i>MemberOf CWE-1000</i> <i>MemberOf CWE-254</i>
<i>CWE-285</i>	Class	Improper Authorization	72	<i>MemberOf CWE-254</i> <i>ChildOf CWE-284</i>
<i>CWE-287</i>	Class	Improper Authentication	2206	<i>MemberOf CWE-1003</i> <i>ChildOf CWE-284</i>
<i>CWE-295</i>	Base	Improper Certificate Validation	585	<i>ChildOf CWE-287</i>

For the set of 163 544 vulnerability instances published in *NVD*, 115 830 instances are assigned threat types by *cvedetails.com*. 47 717 instances, or approximately 29.2% of published instances, have no assigned threat labels, meaning that those vulnerability reports are not categorized to any specific threat category, as presented earlier in Figure 2.6. A vulnerability could be exposed to more than one threat type. For example, vulnerability instance *CVE-2021-39250* is exposed to both *Code Execution* and *XSS*. In total, 90 629 (or 55.4%) vulnerability instances have a single threat-class label, 18 891 (or 11.6%) vulnerability instances have two threat-class labels, 3 704 (or 2.3%) vulnerability instances have three threat-class labels, 2 559 (or 1.6%) vulnerability instances have four threat-class labels, only 4 vulnerabilities are assigned five different threat labels, and only 1 vulnerability instance has six threat-class labels.

Tools like confusion matrix and cross-tab are used to investigate the correlations between threat-labeled vulnerabilities. Based on the confusion matrix, Cramer’s V values are further calculated to quantify the correlations between threat categories. This calculation is conducted using the *scipy.stats.chisq\_contingency* function that computes the chi-square statistic and p-value of independence of variables in a contingency table. Cramer’s V is a formalized version of the chi-square test statistic, with its value in the range  $V \in [0, 1]$ .  $V \in [0.1, 0.3]$ ,  $V \in [0.4, 0.5]$  and  $V > 0.5$  refer to weak, medium and strong associations between the studied variables, separately. The generated Cramer’s V values indicate weak association between any pair of the 13 threat-category vulnerabilities. For example, the strongest correlation among all threat-labeled value pairs is the association between *Overflow* labeled and *Memory Corruption* labeled vulnerabilities, with Cramer’s V equals to 0.289. The weakest associations are between *File Inclusion* labeled and *Directory Traversal* labeled vulnerabilities, and also between *XSS* labeled and *HTTP Response Splitting* labeled vulnerabilities. Both have Cramer’s V equal to 0.

*Code Execution* labeled vulnerabilities have the strongest association with *Memory Corruption* labeled vulnerabilities (with Cramer's V = 0.274), and the least association with *HTTP Response Splitting* labeled vulnerabilities (with Cramer's V = 0.022).

### 5.3 STUDY II: INDUSTRY SURVEY ANALYSIS ON USER EXPECTATION OF VULNERABILITY DATA SOURCES

The purpose of using industry-survey is to gather data from various cybersecurity stakeholders with a background in CI or ICS cybersecurity to collect their feedback and perceptions of the following two questions: (i) the usage of open-source cybersecurity data sources in terms of cybersecurity awareness; and (ii) the usage of vulnerability scoring mechanisms in terms of vulnerability assessment.

This survey-based study was carried out in Sweden, US and China and included 595 participants. We designed the survey as anonymous reports that take about 10 minutes to complete. It is created on an online survey and voting platform that allows survey customization and can be exported as a survey link. The generated questionnaire consists of three main sections to cover information for the aforementioned aspects as well as participant information, which are instantiated through 9 questions to be answered. The design of the questionnaire follows the principles from Somekh and Lewin (2011). The details of all these questions can be found in *Appendix I*. We launch this survey by sending out survey links to potential participants through emails and social media App (e.g. *WeChat* for targeted participants that are located in China). These participants are employed within the areas of CI cybersecurity or closely related fields, such as telecommunication operation and software development. We sent out 1200 invitations or get them distributed through previous contacts. 595 participants submitted their questionnaires, which led to 410 surveys for further analysis after filtering out incomplete ones. This 410 survey set covers respondents located in three countries, namely Sweden (7 participants), the US (3 participants) and China (400 participants). Even though the survey was carried out among a limited population and was a single point measurement so no comparison with another time point, the results of the survey are still valid in gathering users' expectations on publicly accessible vulnerability repositories and vulnerability scoring mechanisms.

Among these 410 respondents, 80 work in ICS security, 89 work in network security, 88 work in IT services, 34 serve in telecommunication operation and equipment maintenance, 38 conduct in cybersecurity-related researches, 41 work in the manufacturing industry, and 40 have close responsibilities in software development.

Figure 5.5 shows the interconnections between respondents' working fields and their frequencies and motivations in using open and public cybersecurity data sources (e.g., *CVE*, *ExploitDB*) and *CVSS*. Generally, participants working with *Telecom Operations/Equipment* and *Software Development* make use of both *CVE* and *CVSS* the least. The factor of Working field has a low influence on the distribution of the reasons in using open and public cybersecurity repositories and standard scoring system *CVSS*. Still, respondents with a career in *Industrial Control System Security*, *Network Security* and *IT Services* show generally higher interests in knowing the general security status. Overall, the strongest two motivations for open and public cybersecurity data sources are knowing the latest security status and comparing and choosing products from different vendors. The latter is also the biggest reason for applying *CVSS* to support cyber vulnerability

assessments.

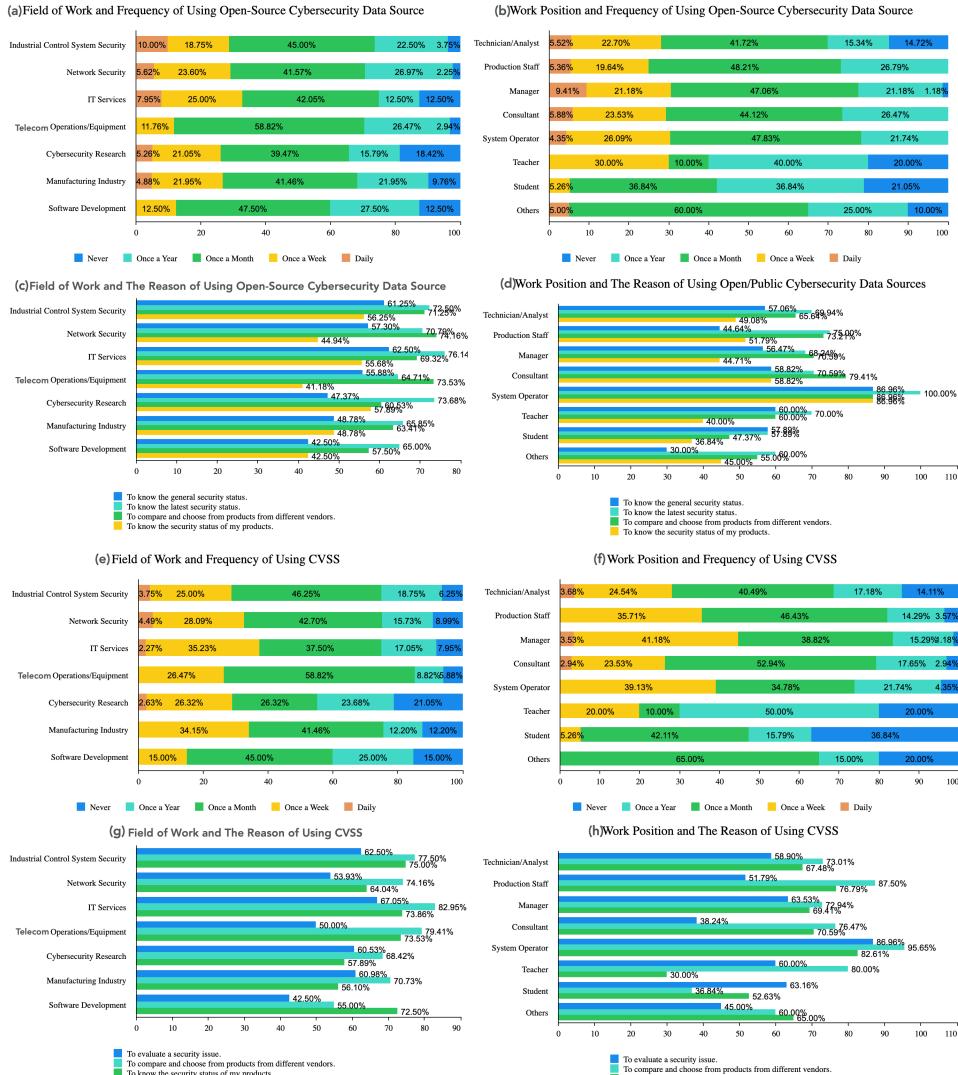


Figure 5.5: Correlation between working field and usage of CVE and CVSS  
(see Figure 10.2 and Figure 10.3 in Appendix I for more details.)

The survey results indicate a moderate usage of open and public cybersecurity data sources among stakeholders. The most desired feature of these cybersecurity repositories with respects to user expectation is to provide the latest security status. More than half of the survey attendants also expect these data sources to deliver complete, accurate and consistent information for products comparison in terms of cybersecurity. CVSS usage is comparatively lower among the survey responders. The users assume CVSS metrics are more likely to support production selection regarding cybersecurity and clarify security status of currently employed products. Even though the survey attendants only

cover a small group of CI cybersecurity stakeholders, the survey results bring inputs to cybersecurity data quality assessment metrics from the users' perspective.

## 5.4 STUDY III: VULNERABILITY DATA QUALITY ASSESSMENT IN ACTUAL CI SYSTEM

This section presents an exploratory case study to validate data quality of online and public vulnerability repositories in terms of data inconsistencies and incompleteness. In this case study, we matched the system scan against a set of public vulnerability databases including *CVE*, *NVD* and vendor security advisories. By doing so, we extracted component-based vulnerability lists for the investigated system, to identify inconsistencies of retrieved vulnerability information. Deep analysis of conflicting vulnerability reports also indicates important insights and guidance for the cybersecurity community on the usage of online public vulnerability data sources. The proposed metrics of data inconsistency analysis are introduced next, followed by a step-by-step illustration of the case study process. Then the systems under investigation are introduced, followed by empirical analysis of the impact of data inconsistencies in the vulnerability-analysis result of the investigated systems.

### 5.4.1 METRIC DEFINITION AND MEASUREMENTS

This case study adopts the definition from Loshin (2010) that data inconsistency refers to data values in one data set not being consistent with values in another data set. Data inconsistency metrics and a set of measurement steps are defined next to assist in identifying and evaluating data inconsistencies in multiple open public vulnerability data sources.

Given a list of  $n$  vulnerability data sources  $[V_1, \dots, V_i, \dots, V_n]$  ( $0 < i \leq n$ ), when querying data source  $V_i$ , we obtain a new set of  $k$  vulnerabilities  $V'_i \subseteq V_i$ . Each vulnerability  $v_{i,j} \in V'_i$  ( $0 \leq j \leq k$ ) has a set of attributes  $a_p \in A$  ( $0 < p \leq m$ ). Therefore, each vulnerability instance  $v_{i,j}$  has a vector  $v_{i,j} = [v_{i,j}^{a_1}, \dots, v_{i,j}^{a_p}, \dots, v_{i,j}^{a_m}]$ . According to Loshin (2010), consistency analysis include 5 contexts, namely record-level, cross-record, temporal, application/business-level, and reasonableness-level consistency. This thesis mainly considers record-level and cross-record data inconsistencies.

#### a) Record-Level Inconsistency

Record-level inconsistency exists between vulnerability attributes  $v_{i,j}^{a_{p1}}$  and  $v_{i,j}^{a_{p2}}$  ( $0 < p1 \leq m, 0 < p2 \leq m, p1 \neq p2$ ). Record-level inconsistency is seen where multiple names are provided to represent the same entity, such as vendor names, vendor-product names, and vulnerable product versions (Anwar et al., 2020). Vendor names are not identical in *CPE* metadata due to various reasons such as misspelled names (e.g., *Schneider Electric* has been spelled as 'schneider-electric' and 'chneider-electric'), and using abbreviated names (e.g., *General Electric Company* has been expressed as 'ge' and 'general-electric'). The causes for inconsistent vendor product names are similar to the ones for vendor names, but is also related to the fact that different stakeholders may provide different names for the same product. Record-level inconsistency appears when different attributes of the same vulnerability provide contradictory information.

### b) Cross-Record Inconsistency

Cross-record inconsistency exists between vulnerabilities  $v_{i1,j}^{a_p}$  and  $v_{i2,j}^{a_p}$  whereby  $v_{i1,j} \in V_{i1}, v_{i2,j} \in V_{i2}$  ( $0 < i_1 \leq n, 0 < i_2 \leq n$ ). This inconsistency indicates scenarios where the same attributes such as vulnerability severity scores, publication dates and vulnerable products are conflicting, or where multiple attributes indicate contradicting vulnerability characteristics.

Different data-repository sources may provide conflicting severity scores. For example, the vulnerability *CVE-2015-6461* is assigned CVSS V2 base score of 5.5 by *NVD* and 3.2 by *ICS-CERT*. The score inconsistency arises due to contradictory conclusions on the access vector of this vulnerability, for which *NVD* assigns it as network-based and *ICS-CERT* assigns it as local-based. Under *CVE-2017-6023*, this vulnerability is assigned CVSS V3 base score of 9.8 with vector *AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H* by *NVD* and 7.3 with vector *AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L* by *ICS-CERT*, separately. A more general example is shown in Figure 5.6. Considering a random vulnerability instance  $v$ , *NVD*, the corresponding manufacturer, and a third-party analyst provide their severity scores as  $S^{NVD}$ ,  $S^{Manufacturer}$  and  $S^{Analyst}$  which can be inconsistent. Despite CVSS popularity (Scarfone and Mell, 2009) (Fang et al., 2020), inconsistency among reported scores for the same vulnerability instance does occur. Particularly, when considering other CVSS temporal and environmental metrics, vulnerability properties evolve across time and deployment environments. Hence, additional sources of relevant data, including manufacturer-provided data as well as online reviews from relevant security sources and forums, are expected to consolidate further existing CVSS scores (Johnson et al., 2016b).

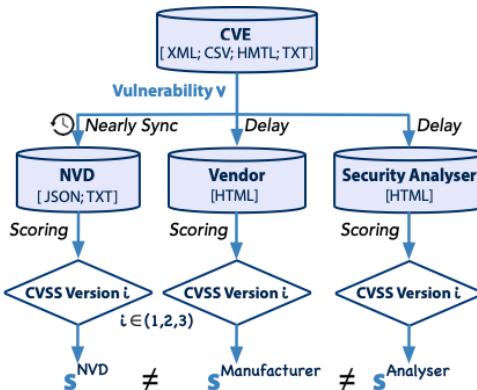


Figure 5.6: Potential time delay of scoring and inconsistent scores

Publication dates indicate the period when vulnerabilities become public and provide essential tracks for system protection prioritization. Nevertheless, inconsistencies of such publication dates occur commonly due to disagreement between *NVD* publication dates and public disclosure dates on other vulnerability databases and cybersecurity blogs (Anwar et al., 2020) (Jo et al., 2020).

*CPE* vulnerable version ranges for each *NVD* entry can be inconsistent with identified vulnerable versions from third-party analysts and vendors (Dong et al., 2019). One example is *CVE-2019-5527* that refers to a use-after-free vulnerability in the virtual sound device. According to the security advisory *VMware*, mitigation for this vulnerability is to upgrade the vulnerable component *VMware ESXi* to *ESXi670-201904101-SG* (this is a

patch package name and is linked with build number 13006603). And hence, *VMware ESXi* versions later than *ESXi670-201904101-SG* are not affected by this vulnerability. However, those later versions like *ESXi670-202004001* are listed as vulnerable versions in *NVD*.

#### 5.4.2 DATA INCONSISTENCY ANALYSIS PROCESS

The process of data inconsistency analysis is composed of four main steps, as illustrated in Figure 5.7.

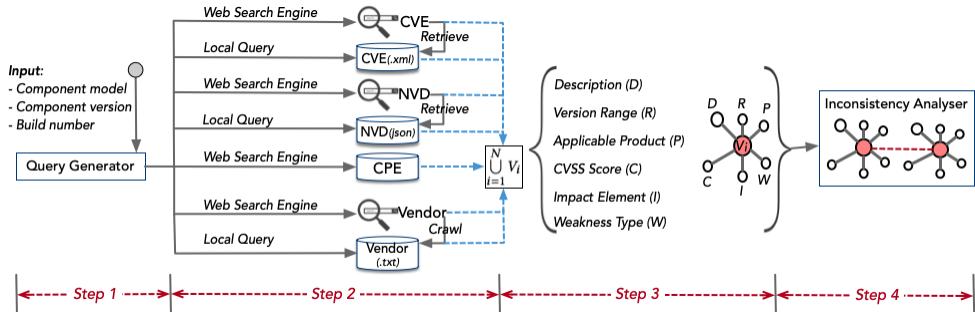


Figure 5.7: Overview of the data inconsistency analysis process

##### Step 1: system configuration information collection and query tags generation

The first step is to collect system configuration information and component details. We programmed an import filter to translate component information (usually documented in a CSV file) into query tags. Data items like models, versions and the latest patching update (e.g. cumulative security update KB, or knowledge base, package numbers) are necessary for vulnerability retrieval from online public vulnerability data sources. For instance, given description “*Trend Micro ServerProtect for Storage 6.0*”, the generated query tags are *type: software, vendor: Trend Micro, product: ServerProduct, vulnerable function: Storage, version: 6.0* in a format similar to the *CPE* match-list metadata.

##### Step 2: extract component-based vulnerability instances

Secondly, the translated tags from the previous step were used in Python-based queries in the PC terminal to extract component-based vulnerability instances. We then matched these tags against the *CPE* metadata to retrieve corresponding vulnerability instances from *NVD*. We also converted the downloaded disk-based JSON files from *NVD* into a Python Dictionary data structure that organizes the data as key-pair values. Some values are scalars (e.g. *CVSS* entries), while others can be a list of other dictionaries (e.g., *CPE* entries). The query tags are then matched against *CPE* entries of each vulnerability instance and return the instances that contain these tags. The description entry of *CVE* JSON files is parsed to extract matching vulnerability instances. This manner is further composed of multiple traditional text processing sub-steps such as tokenization and removing stop-words (Zhu and Dumitras, 2016), as well as word matching. We applied similar data prepossessing process to crawled vendor reports in CSV or JSON format, while using two ways to fetch vulnerability reports from vendor websites. The first approach refers to locating the vendor-related URL references of disclosed vulnerabilities in *CVE* through reference maps, based on which the contents of the referred websites

are fetched. The other method is that, given a vendor product, the vendor website is requested to fetch vulnerability-related data. The union of all retrieved vulnerability reports  $\bigcup_{i=1}^N \{v_{i,j}\}$ , or  $V_{CVE} \cup V_{NVD} \cup V_{vendor}$  is generated as the results.

#### Step 3: extract attributes for each vulnerability instance

In the third step, the attributes listed in Table 5.4 are extracted for each vulnerability instance. By doing so, one vulnerability instance has a vector of six attributes  $v_{i,j} = [v_{i,j}^{a_1}, \dots, v_{i,j}^{a_p}, \dots, v_{i,j}^{a_6}]$  ( $0 < p \leq 6$ ), where each attribute has a list of up to 3 values. We directly parsed retrieved vulnerability reports to obtain the targeted attributes like affected products and version ranges.

#### Step 4: assess inconsistencies of vulnerability data sources

The last step focuses on the comparison between the attribute sets of each vulnerability instance set, in order to diagnose inconsistencies. The statistic patterns are extracted that are based on vulnerability instances retrieved from different data sources, to assess the degree of data inconsistencies. Finally, the impact of data inconsistencies of multiple online vulnerability data sources is evaluated in supporting vulnerability analysis.

### 5.4.3 INVESTIGATED CI SYSTEM

The information regarding the investigated system is collected through 3 interviews and also documents shared by the system owners.

The investigated IT system is composed of 3 major sub-systems, namely a DataCenter, an application layer and a network layer, as depicted in Figure 5.8. DataCenter integrates hardware components like physical servers with operating system software. DataCenter contains 14 components, including a middle-ware server that bridges multiple partners' systems and an USBHub server that works as a USB network gate. Additionally, a hypervisor host deploys and serves virtual systems, which provides an abstraction layer for virtualization. This virtualization layer supports multiple hypervisors (or virtual machines), together with heterogeneous operating systems and applications that run in isolation.

The application layer includes the configuration of 3 system packages supported by hypervisors, namely the IoT system (containing 138 components), customer management system (containing 137 components), and control system (containing 27 components). Among these three systems, the control system enables monitoring and controlling the physical system. The IoT system records physical process data and then sends collected information to the customer management system through the middle-ware server. The customer management system stores, analyses, and manages consuming data. In an IoT system, for example, 4 hypervisors are included to integrate guest operating system, access control (AC) server, encryption key-store (EKM) server, database (DB) server and application (APP) server. Note that servers within the same security zone are reachable to each other. Servers in different zones can communicate through specific access lists.

Physical servers in the DataCenter are connected to switches through fibers. The network layer represents the IT network's wireless connections that contain 7 components. Some of the critical components are firewall, network operating system and wireless controller. These switches are further connected to the OT servers. The OT system comprises 16 components that collect information and directly monitor physical processes.

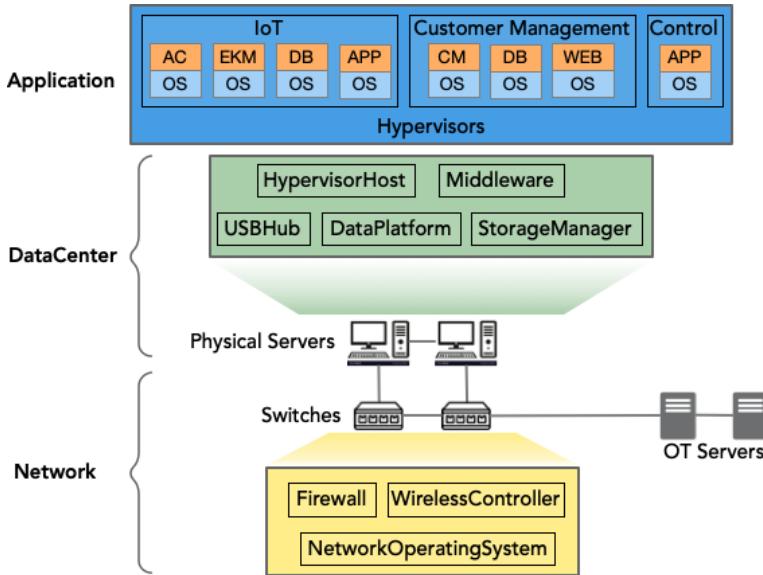


Figure 5.8: Structure of the investigated IT and OT systems

#### 5.4.4 DATA SOURCES

This case study aims to observe and to compare inconsistencies between collectable data from *CVE*, *NVD*, and vendor repositories. Three main types of vulnerability repositories are queried to retrieve related vulnerability instances of the investigated system, namely *CVE* (or  $V_{CVE}$ ), *NVD* (or  $V_{NVD}$ , and include *CPE* entries), and Vendor websites (or  $V_{vendor}$ ). This case study focuses on six attribute sets for vulnerability analysis that are available in a minimum of two data sources, as depicted in Table 5.4. Description typically covers the applicable product, version range, the vulnerable function, and sometimes the weakness type. *CVE*, *NVD*, and vendor websites provide access to such vulnerability descriptions. The applicable product and version range are identified and extracted from the *CPE* metadata or corresponding sections of vendor websites.

Table 5.4: Deployed vulnerability data sources in *Study II*

Attribute	<i>CVE</i>	<i>NVD</i>	Vendors
Description	Yes	Yes	Yes
Applicable Product	Yes	Yes	Yes
Version Range	Yes	Yes	Yes
CVSS Score	N/A	Yes	Some
Impact Element	Some	Yes	Some
Weakness Type	Some	Yes	Some

### 5.4.5 ANALYSIS OF RETRIEVED VULNERABILITIES

This case study focuses on cross-record data inconsistencies, especially between *NVD* and vendor advisories. CNA discloses vulnerabilities further analysed by *NVD*, vendor and other security analysts. *CVE* reports are deployed to provide ground truths in some studies that perform inconsistency detection (Dong et al., 2019). In contrast, some other studies employ *NVD* entries as ground truths (Nappa et al., 2015). Nevertheless, the assumption of *CVE* or *NVD* as ground truth may not be valid. Instead, the purpose of this study is to explore the impact of relying on a singular data source for vulnerability analysis, especially between *NVD* and vendor repositories.

Investigated System	Component Amount	<i>CVE</i>	<i>NVD</i> (CPE)	Vendor	Union	Intersection
IoT System	138	75	77	69	77	69
Customer Management System	137	309	319	311	319	309
Control System	27	50	53	28	53	28
OT	16	5	5	25	25	5
DataCenter	8	18	18	13	18	13
Network	7	105	75	11	105	11

Table 5.5: Retrieved vulnerability instances in *Study II*

The investigated system contains components provided by 18 different vendors, among which ten vendors provide specific sections for security advisories. These ten vendors also provide online security advisories to address vulnerabilities. For the remaining eight vendor data sources, only three vendors have identifiable vulnerability records in *CVE* or *NVD*. The distributions of vulnerability instances in different systems are listed in Columns 3-5 in Table 5.5. The network has comparatively more vulnerability instances per component, while DataCenter has the least vulnerability instances per component. 562, 536, and 457 instances are separately extracted from *CVE*, *NVD*, and vendors. The number is 597 when taking the union of all the identified vulnerabilities, which changes to 435 when taking the intersection of the vulnerabilities, as listed in Columns 6 in Table 5.5. Except for OT related vulnerabilities, the union of vulnerabilities retrieved from *CVE* and *NVD* cover the vulnerabilities extracted from the vendor websites. This observation only applies to this case study as the involved vendors are limited. On average, the customer management system has the highest number of vulnerability instances per component in the intersection set. In contrast, the OT system has the least vulnerability instances per component in the intersection set.

The inconsistencies between different vulnerability report sources are reflected in the affected products and version ranges, weakness categorization, exploit correlation, and vulnerability severity.

### 5.4.6 DATA INCONSISTENCIES IN AFFECTED PRODUCTS

Due to software upgrade versioning, version numbers directly extracted from computer specifications are usually release versions. Yet, release versions may not reflect the installed update-package numbers of the component or the internal-version numbers from

the vendors. One example is the *VMware vCenter Server* with internal version numbers from the earliest version *virtualCenter 2.5.0 GA* to the latest version *vCenter Server 7.0 update 1d (7.0.1.00300)*, till March 8, 2021. The release version *vCenter Server 6.0* has 18 different internal version numbers, each of which is related to various release updates.

The impact and cause of discrepancies in terms of affected products and version ranges are reflected in the different amounts of identified vulnerabilities in Table 5.5. The network has the highest rate of inconsistent vulnerabilities in the IT system, followed by the IoT and control systems. These discrepancies mainly result from inconsistent product names and different views on whether a specific component version is vulnerable. This indicates an underlying problem of synonyms of the existing cybersecurity repositories, which further suggests a need for a better mechanism to identify vulnerable products by unified names. Table 5.6 lists the vulnerability instances found in *NVD* and vendor entries, separately.

Table 5.6: Data inconsistencies in affected products and version ranges between *NVD* and Vendors

IoT System	Component Number	Consider NVD Only		Consider Vendor Entries Only	
		Vulnerability	Component with The Most Vulnerabilities	Vulnerability	Component with The Most Vulnerabilities
Operating System	3	21	Directory Component	12	Database Tool-set Component
Access Control (AC) Server	16	48	Anti-malware Component	28	Run-time Library Component
Encryption Key-store (EKM) Server	36	75	SDK Component	55	Run-time Library Component
Database (DB) Server	40	66	Database Management Component	46	Run-time Library Component
Application (APP) Server	43	74	Database Management Component	46	Run-time Library Component
Customer Management System	Component Number	Consider NVD Only		Consider Vendor Entries Only	
		Vulnerability	Component with The Most Vulnerabilities	Vulnerability	Component with The Most Vulnerabilities
Operating System	4	20	Remote Desktop Component	19	Remote Desktop Component
Customer Management (CM) Server	64	253	Office Application Component	232	Office Application Component
Database (DB) Server	52	68	SDK Component	48	SDK Component
Web (WEB) Server	17	44	Run time Library Component	17	Run-time Library Component
Control System	Component Number	Consider NVD Only		Consider Vendor Entries Only	
		Vulnerability	Component with The Most Vulnerabilities	Vulnerability	Component with The Most Vulnerabilities
Operating System	4	12	Directory Component	8	Directory Component
Application (APP) Server	23	65	Database Management Component	45	Run-time Library Component
Other Systems	Component Number	Consider NVD Only		Consider Vendor Entries Only	
		Vulnerability	Component with The Most Vulnerabilities	Vulnerability	Component with The Most Vulnerabilities
Data Center	8	19	Hypervisor Component	11	Hypervisor Component
Network	7	75	Network Operating Component	11	Firewall Component
OT	7	5	Open Platform Communications Component	25	Protection and Control Component

The component with the most vulnerabilities of each investigated system differs when involving different vulnerability data sources:

- The communication network system has the top data inconsistency rate among the six investigated sub-systems, followed by the OT system. For example, in the OT system, the open platform communication (OPC) component has the highest amount of vulnerabilities when considering only *NVD* entries. When taking into consideration only the vendor entries, the protection and control component contributes the most vulnerabilities.

- In the IoT system, the EKM server contributes the highest amount of vulnerabilities. The five sub-systems of IoT system, namely operating system, AC server, EKM server, DB server, and APP server, have different rankings of the most vulnerable component after considering data inconsistencies. For instance, the database tool-set component has the most vulnerabilities in the operating system, when considering only vendor entries. When accounting only *NVD* entries, the most vulnerable component in the operating system is the directory component.
- In the customer management system, the web server contributes an enormous amount of inconsistent vulnerabilities. The inconsistencies between *NVD* and vendor entries in the vulnerability sub-sets of customer management sub-system do not affect which component has the most significant amount of vulnerabilities in each sub-system.
- In the control system, the application server has a higher rate of inconsistent vulnerabilities compared to the operating system. When accounting only entries from the vendor, the run-time library component has the most vulnerability instances in the application server, which is rectified to the database management component when accounting only entries from *NVD*.

#### 5.4.7 DATA INCONSISTENCIES IN WEAKNESS TYPE AND CVSS IMPACT

The inconsistencies between vulnerability report sources are reflected in weakness categorization, impact evaluation, and vulnerability severity. We visualize such discrepancies in several graphs and discuss them next.

Figure 5.9 visualizes the influence of vulnerability data inconsistencies on exploitability and impact levels of the whole investigated system. A decision with only *NVD* entries as references may guide cybersecurity analysts to pay closer attention to threats with local path based exploits, low access complexity, and no authentication requirement. Meanwhile, the analysis result based on *NVD* entries suggests that the investigated system is suffered from a higher impact on confidentiality, integrity and availability.

The *CVSS* version 2 base-scores of the investigated systems are examined to check if inconsistent vulnerabilities affect the average severity of these investigated systems. As depicted in Figure 5.10, the customer management system and IoT system have similar vulnerability severity score distributions no matter which cybersecurity data source is used. In contrast, vulnerabilities from *NVD* entries contribute more vulnerabilities with higher severity scores (equal to or larger than 7) to Data Center, Network, and Control System. This is particularly true for the communication network-related vulnerability instances, whereby a large ratio of *NVD* vulnerability entries has a *CVSS* V2 base score between 7.0 and 8.0. Another interesting observation is that vulnerabilities of the OT system found in *NVD* contribute to lower *CVSS* scores compared to the instances found in the vendor sites. To summarize, most discrepancies are found in *Network* and *Control System* vulnerabilities.

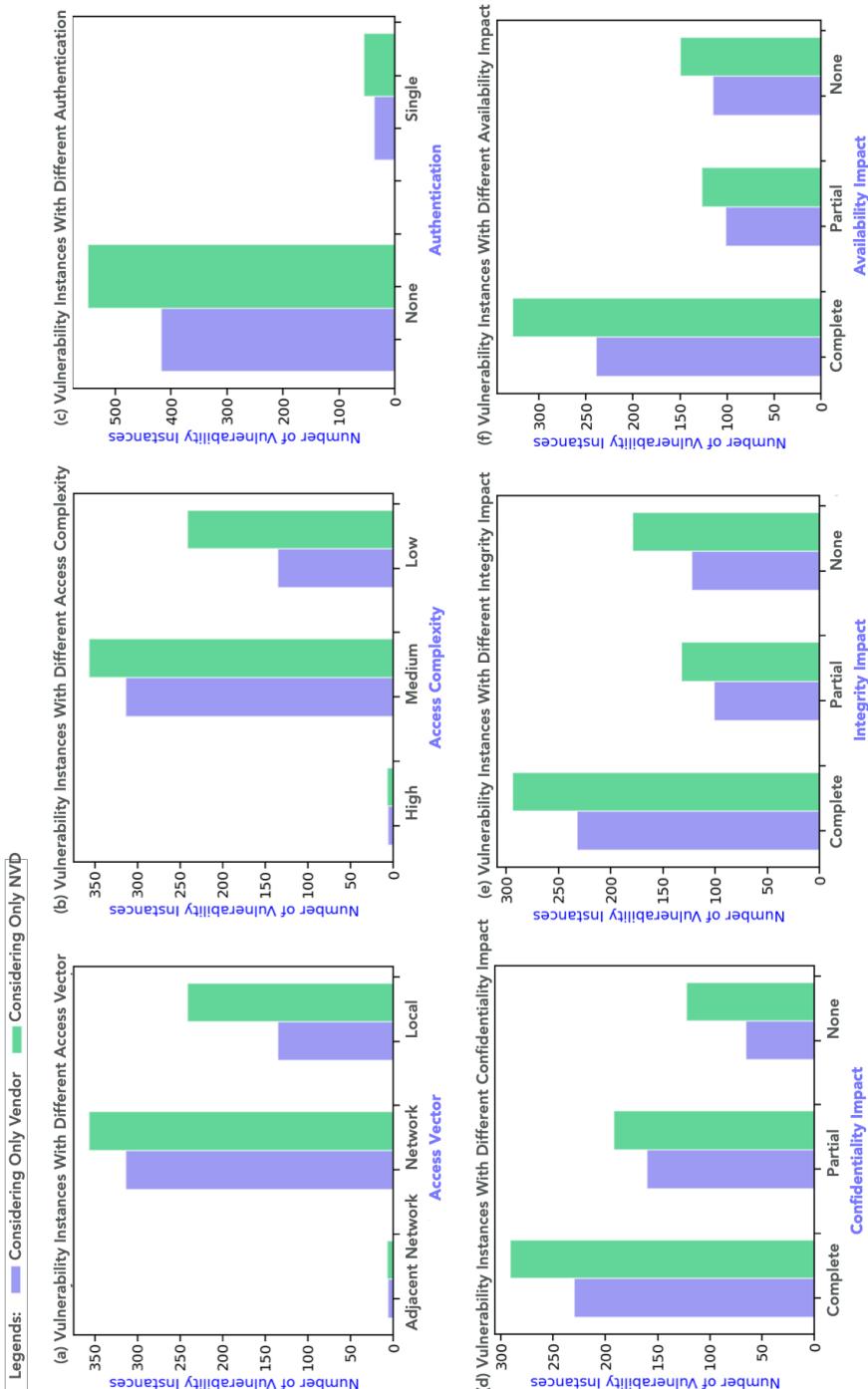


Figure 5.9: Data inconsistencies in exploitability and impact between NVD and Vendor. Exploitability and impact analysis are based on the assigned CVSS labels.

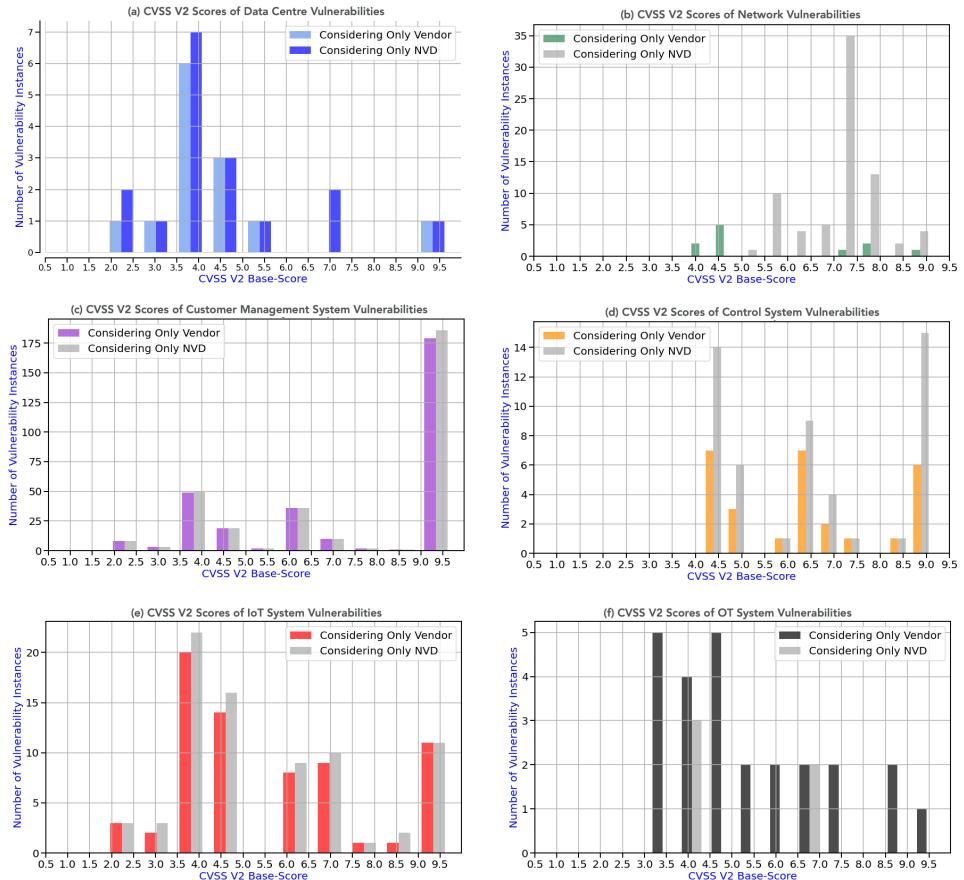


Figure 5.10: Data inconsistencies in CVSS V2 scores between NVD and Vendor.

Inconsistent vulnerability data sources also have an impact on patch guidelines of the major weaknesses exemplified by the weakness enumerations *CWE* of extracted vulnerabilities. As shown in Table 5.7, the top 3 weaknesses with the highest occurrence frequencies in all the investigated subsystems are partially changed or revised with rankings, except for the control system. For instance, the most frequently appeared network weaknesses when only accounting vendor entries are (i) exposure of sensitive information to an unauthorized actor, and (ii) incorrect permission assignment for the critical resource. Among these two network weaknesses, the first weakness occurs due to direct or indirect insertion of sensitive information, such as system environment, network status and configuration, intellectual property, and private customer records, into resources accessible to unauthorized actors. Excavation and fingerprinting attacks may be triggered to exploit the information leak. Possible mitigations against this weakness include encryption and password-protection of sensitive data and appropriate compartmentalization that reinforces privilege separation functionality. The second network weakness refers to incorrect permission assignment for critical resources, which impacts the confidentiality and integrity of sensitive properties once successfully exploited through attacks like signing malicious code. Environment hardening is one mitigation for proper

permission assignment. The remaining network weaknesses that appear only once are not listed in Table 5.7. Taking *NVD* as the only data source, then the major network weaknesses are revised as (i) improper input validation, (ii) improper neutralization of unique elements used in an OS command, and (iii) uncontrolled resource consumption. These ambiguous weaknesses may address more attention towards protecting the system against code execution, DoS, bypass protection mechanism, and other similar cyber threats, resulting in less budget in mitigating the existing weaknesses.

Table 5.7: Data inconsistencies in the assigned *CWE* between *NVD* and Vendor

Investigated System	Most Frequently Appeared Weaknesses Consider <i>NVD</i> Only	Most Frequently Appeared Weaknesses Consider Vendor Entries Only
DataCenter	<ol style="list-style-type: none"> <li>1. Improper use of previously-freed memory.</li> <li>2. Improper certificate validation.</li> <li>3. Exposure of sensitive information to an unauthorised actor.</li> </ol>	<ol style="list-style-type: none"> <li>1. Improper certificate validation.</li> <li>2. Insufficiently protected credentials.</li> <li>3. Improper use of previously-freed memory.</li> </ol>
Network	<ol style="list-style-type: none"> <li>1. Improper input validation.</li> <li>2. Improper neutralization of special elements used in an OS command.</li> <li>3. Uncontrolled resource consumption.</li> </ol>	<ol style="list-style-type: none"> <li>1. Exposure of sensitive information to an unauthorised actor.</li> <li>2. Incorrect permission assignment for critical resource.</li> </ol>
IoT System	<ol style="list-style-type: none"> <li>1. Improper privilege management.</li> <li>2. Improper permissions and access controls.</li> <li>3. Improper input validation.</li> </ol>	<ol style="list-style-type: none"> <li>1. Improper privilege management.</li> <li>2. Improper input validation.</li> <li>3. Improper permissions and access controls.</li> </ol>
Customer Management System	<ol style="list-style-type: none"> <li>1. Improper restriction of operations within the bounds of a memory buffer.</li> <li>2. Improper input validation.</li> <li>3. Exposure of sensitive information to an unauthorised actor.</li> </ol>	<ol style="list-style-type: none"> <li>1. Improper restriction of operations within the bounds of a memory buffer.</li> <li>2. Exposure of sensitive information to an unauthorised actor.</li> <li>3. Improper input validation.</li> </ol>
Control System	<ol style="list-style-type: none"> <li>1. Improper neutralization of input during web page generation.</li> <li>2. Improper control of generation of code.</li> <li>3. Improper permissions and access controls.</li> </ol>	<ol style="list-style-type: none"> <li>1. Improper neutralization of input during web page generation.</li> <li>2. Improper control of generation of code.</li> <li>3. Improper permissions and access controls.</li> </ol>
OT	<ol style="list-style-type: none"> <li>1. Incorrect permission assignment for critical resource.</li> <li>2. Improper privilege management.</li> </ol>	<ol style="list-style-type: none"> <li>1. Out-of-bounds Read.</li> <li>2. Out-of-bounds Write.</li> <li>3. Incorrect permission assignment for critical resource.</li> </ol>

#### 5.4.8 EVALUATION THROUGH INTERVIEWS

The interview-based evaluation included here was performed concurrently with this study. It should be noted that this evaluation study differs from the three interviews performed for collecting system information. This study composes of two interviews at the beginning and the end of *Study II*. The objectives of these two interviews are three-fold: (i) to know how the vulnerability assessment results relate to security protection in the organization; (ii) to analyze the quality of cybersecurity decisions made following the identification of data inconsistency issues; and (iii) to gain further knowledge on the current usage of available cybersecurity data sources. We prepared a list of questions before

these interviews, of which more details are presented in the *Appendix II*. These interviews are conducted in a semi-structured iterations. This means we read and analyzed interview transcripts early, rather than waiting until all interviews have been completed. Meanwhile, we modified and added follow-up questions based on these early analytic results. The interview iteration also opens the opportunity to further discussions related to CI cybersecurity and data quality requirements.

We interviewed three cybersecurity experts that are employed in the investigated organization or a cooperated stakeholder in Sweden to ensure the confidentiality of the studied system and vulnerability data. These three experts cover the roles of IT administrator (quoted as *Expert A*), IT security analyst (quoted as *Expert B*) and OT security operator (quoted as *Expert C*). They hence have in-depth knowledge of the cybersecurity needs and actual status of the investigated system.

The following text is a summary of some key points obtained from the interviews, grouped by subjects. Comments not related to the subject have been omitted. When multiple people pointed out the same, the mentioned topic is included as one point.

#### (i) Vulnerability Assessment Brings Valuable Insights

Vulnerability assessment in this case study delivers a significant number of vulnerability reports using public vulnerability repositories, and generates a broad picture of the vulnerable level of the whole system. *Expert B* expressed that “*Before the case study, we had detailed and up-to-date information of some vulnerabilities, but mainly for those regarding critical assets. This case study gives us a more comprehensive picture of the vulnerable status of the system*”. Vulnerability management of a complex and large-scale IT/OT infrastructure is challenging with respects to gain a full and up-to-date overview of the vulnerability situation. This is particularly true for offensive security works like penetration testing and some defensive security inspections. This challenge is exemplified by the fact that many of today’s OT systems have most of their cybersecurity-related tasks outsourced to other stakeholders, as quoted from *Expert C* that “*We cooperate with and rely upon some institutions for their IT services and IT security*.”.

Valuable references cover unknown weaknesses that fit the gap of vulnerability management. This is exemplified by quotations from *Expert A* that “*Some vulnerabilities in the embedded software of the customer management system and control system were unknown*”. “*Office application components are regarded as low criticality, and do not address enough attention in this organization’s cybersecurity perspective*.” from *Expert A* suggests that the diagnosed substantial amount of vulnerability instances in the office application components is not expected. These vulnerability instances also have a high proportion of critical severity scores.

#### (ii) Diversity, Incompleteness and Inconsistencies of Cybersecurity Databases

The main objectives of using open-source security databases are in line with the survey results in the previous section, namely to know the latest security status as well as to compare and choose more secure products. One underlying challenge is to manage diverse security alert sources. *Expert B* stated that “*We get informed of the latest security status mainly by subscription to the vendors. Some vendors send out monthly vulnerability reports and patch suggestions to the customers. There are also vendors that invite customers to private meetings for vulnerability alert discussions, before publishing relevant vulnerabilities*”. Such a subscription normally leads to hundreds or even thousands of security reports to be reviewed. Manual auditing and checking vulnerabilities that are relevant to their systems are demanding. *Expert A* addressed that “*We*

*want both general security status and detailed information about vulnerabilities only relevant to our system”.*

Some known vulnerabilities in the OT system cannot be found using online cybersecurity repositories and are not published by the vendor. The lack of OT vulnerability records also indicates the incompleteness of online cybersecurity data sources. *Expert C* explained that “*We were notified of some OT vulnerabilities through vendor emails, yet those vulnerabilities are not seen in the case study results*”. *Expert B* added that “*We mostly use vendors through subscriptions. Besides that, subscription of the national CERT source brings extra value*”.

Data inconsistencies are not uncommon for vulnerabilities’ affected products and version range. However, “*These data inconsistencies bring a higher level of uncertainty in cybersecurity decision making with amendments in the ratios of major weakness and threat types, and can be misleading in priority patching.*”, according to *Expert B*. Nevertheless, consistent data and indicators reassure conscientious and efficient patch actions given a certain budget. This need is in compliance with the practices that “*Some vulnerabilities need to be patched immediately considering severity and exploit likelihood. Some non-critical vulnerabilities are patched following a monthly schedule. PC-based vulnerabilities need to be patched normally within 2 weeks. There are also vulnerability alerts that are related to critical enterprise processes, and are therefore hard to be patched.*”, quoted from *Expert A*.

### (iii) CVSS Sub-Vectors are Valuable

Cybersecurity decision making is less dependent on CVSS scores, as quoted by *Expert B* that “*The mathematical score itself is not so meaningful. But the sub vectors, such as the attack vector and confidentiality impact, are helpful in terms of root-cause analysis and patch guiding.*”. *Expert A* points out that “*We need indicators to know the security status of our products.*”

## 5.5 CONCLUSION

This chapter starts with exploratory studies to investigate the characteristics of public accessible vulnerability repositories. The variety of employed syntaxes of vulnerability taxonomies suggests the heterogeneity of these repositories. *CVE* entries are used as the sources for further vulnerability feature assessments. For example, *NVD* assigns *CVSS* severity scores and *CWE* weakness categories to *CVE* unstructured data, and generates structure reports accordingly. *CWE* and *CAPEC* also map some of their entries to *CVE* records. We also included a static analysis (*Study I*) of *NVD* historical records retrieved at November 3, 2021. Results of this study spotlights the diversity of *NVD* data. Our static findings indicate some existing challenges in using public vulnerability data sources, namely diversity and heterogeneity. The baseline study addresses the general challenges in correlating and synthesizing diverse and heterogeneous data sources for vulnerability assessment.

To understand better the specific challenges aligned with CI cybersecurity stakeholders, we carried out a survey study (*Study II*) to collect these stakeholders’ preferences in terms of using public vulnerability data sources and scoring standards like *CVSS*. More than 70% of the 410 respondents embrace using these data repositories to know the latest security status as well as to compare and choose more secure products.

We further carried out a vulnerability data quality assessment case study (*Study III*) using actual system configuration information from a data center located in Sweden. Re-

sults of this case study indicate that cybersecurity budget allocation may differ by relying on only NVD vulnerability entries or vendor security bulletins. During this case study, many vulnerability instances are extracted that contain some unexpected weaknesses that decision-makers were unaware of before. The influence of inconsistent vulnerabilities is systematically measured in exploitability, impact levels, and the primary weakness types of the investigated system. This case study also showed that the number of inconsistencies could vastly differ depending on the type of software component. The high rate of inconsistent vulnerabilities in the control system and the network components are only valid in this case study. The rate of data inconsistency may differ in other organizations. Still, specific sub-systems can render a high percentage of data inconsistency.

Query generation, vulnerabilities retrieval, and data inconsistencies evaluation were conducted in a semi-automated manner in the studies included in this chapter. Specifically, query tags are generated through both automatic terminology extraction and manual check. These query tags were matched against vulnerability data sources to obtain related vulnerability instances. The inconsistent vulnerabilities identified in *Study II* are manually checked to filter out instances that do not apply to the system. The whole case study process took around four weeks to complete, which may leave a gap between a vulnerability exploit occurrence and the deployment of an available patch. This is especially true considering that some vulnerabilities need to be patched immediately or within two weeks, according to our follow-up interviews after the case study. The root causes of the inconsistencies are not investigated, i.e., which vulnerability instance was created first, when it was updated, when or if duplicates were detected, and so forth. However, these studies indicates that better workflows should be installed to synchronize and correct vulnerability instances in the public and vendor repositories.

In summary, the systematic integration of vulnerability instances into a kind of data warehouse offers the opportunity to detect inconsistencies and refer them to the maintainers of the original sources. Furthermore, vulnerabilities are reported in various standards, which increases the difficulty of vulnerability-related provisioning and sharing. Vulnerability reporting standardization is one way to enhance the quality of shared cyber threat intelligence. An approach that correlates various vulnerability data sources and deploys trustworthiness data verification brings substantial benefits to vulnerability identification and management.



# VULNERABILITY DATA CORRELATION AND CONSOLIDATION



## CHAPTER 6

# VULNERABILITY DATA CORRELATION AND CONSOLIDATION

The data population size (nearly 1.68GB for baseline studies) and the data quality issues detected in Chapter 5 make it respectively feasible and desirable to devise an integrated database for the purpose of vulnerability analysis in a CI company.

However, when it comes to collecting, managing, and correlating vulnerability data from a wide range of sources, several challenges arise. Some typical characteristics of vulnerability repositories, such as heterogeneity (Christey and Martin, 2013) and information gaps (unreported vulnerability attribute instances) (Ladd, 2017), bring obstacles to employing these repositories for accurate assessments. The information regarding the same vulnerability instance is expected to be integrated into a standard cross-linked structure to support accurate vulnerability analysis (Mavroeidis and Bromander, 2017), as discussed in Section 3.1. A thorough vulnerability assessment requires the analysis of heterogeneous data streams to find interconnections between dependent vulnerability concepts. These standalone objects of vulnerability, attack, and threat are currently stored in separated standardized enumerations (Dong et al., 2019). Some of these object pools have cross references with each other, but still lack thorough analysis of mapping intra-features and inter-features (Christey and Martin, 2013). At the same time, some object pools do not include marking information. Furthermore, one object may be defined in more than one standalone enumeration, which increases the difficulty of filtering irrelevant information about this object. Nevertheless, further research that incorporates heterogeneous data from different databases while providing structured and simplified indicators is limited.

This chapter aims to answer the question of how to obtain and correlate data for vulnerability analysis, considering complex and heterogeneous sources of security alerts. To investigate this question, we propose an approach that correlates various cybersecurity data sources and converts these disparate repositories into a common data model (CDM) to bring substantial benefits to vulnerability identification and management. An instantiation of the proposed CDM against 12 widely used cybersecurity data sources and enumerations. Data obtained by crawling several security-alerts from online repositories as well as manufacturer websites is condensed into relevant views in this localized and synchronized database, and is assisted with a *Query Method* to retrieve vulnerability instances.

The suitability of this suggested CDM is validated through practical vulnerability assessment use cases that emerge from this instantiated databases. Two studies are included in the validation studies: *Study IV*, in which we evaluate the feasibility of the proposed CDM using general CIs vulnerability trend analysis; and *Study V*, in which we validate the performance trade-offs of the proposed approach in comparison to other similar works.

This chapter covers, but is not limited to results from *Paper II* and *Paper V*.

## 6.1 ARTIFACT I: VULNERABILITY DATA MODEL AND QUERY GENERATION METHOD

The structure of our proposed cross-linked and correlated vulnerability database is presented in this section, including the common data model and integration schema.

### 6.1.1 OVERALL STRUCTURE AND COMMON DATA MODEL

In data warehouse based studies, the design and maintenance of ETL (refers to extraction, transformation, and loading) processes are key factors (Luján-Mora, Vassiliadis, and Trujillo, 2004). ETL processes are divided into extraction of data from heterogeneous data sources, transformation of data such as conversion and cleaning, and data loading into a data warehouse.

We correlate vulnerability instances with information related to affected systems, considering attributes such as weakness, threat exploits and potential attack scenarios. These data objects were categorized and discussed earlier in Table 5.1 in Chapter 5. Figure 6.1 illustrates briefly the correlation between vulnerability instances and relevant security information extension, whereby vulnerability instances can be retrieved from official databases and also from manufacturer websites as well as other online sources such as forums and blogs. Furthermore, cross references are collected from multiple repositories leading to standardized enumerations, using common keys like *CVE ID* as a unique vulnerability identifier.

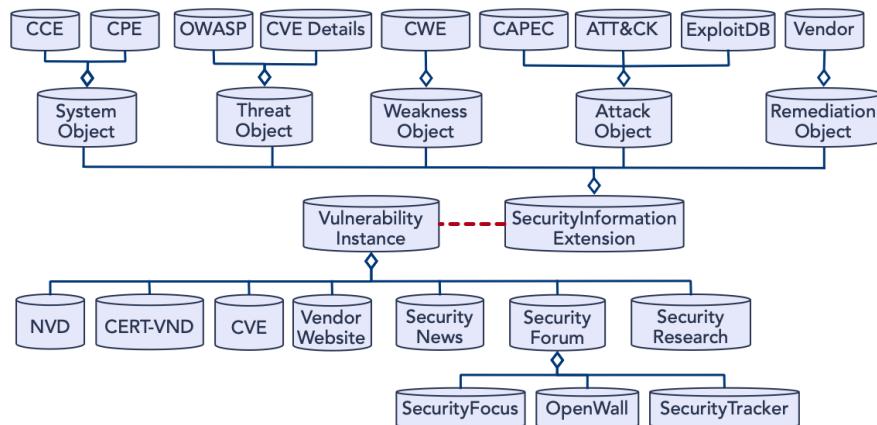
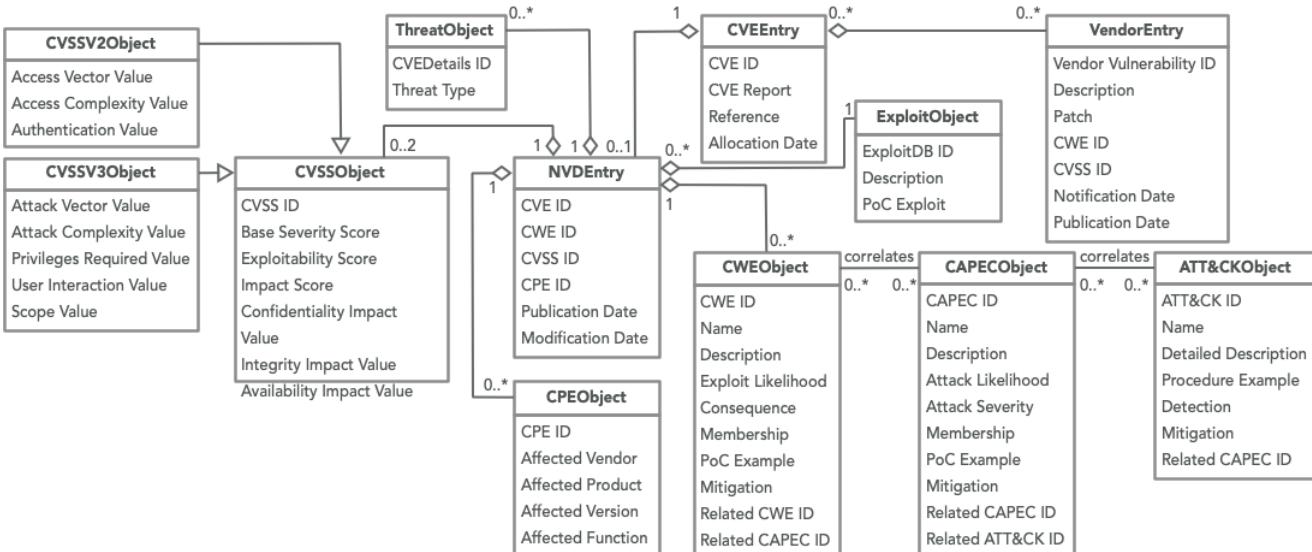


Figure 6.1: Bird view of vulnerability database overall structure

We further created a detailed semantically-grounded data model to ensure a clear mapping of the data sources to our data warehouse structures. Besides the source conceptual schema diagram of Figure 6.2, this section contains several data mapping diagrams as complementary documentations to capture the data flow and correlations of the involved data sources, each of which is defined at a different level of abstraction.

Figure 6.2 presents the proposed CDM model in an Unified Modeling Language (UML) representation to outline the sources that feed our data warehouse and their attributes.

Figure 6.2: Common data model: source conceptual schema.  
(Note that this is a simplified version to ensure readability)



We defined two types of data classes, namely data entry class (e.g. *NVDEntry* and *VendorEntry*) that is usually a source of vulnerability instances, and data object classes (e.g., *CWEObject* and *CVSS Objects*) that is usually a language or evaluation metric for vulnerability assessment. And hence, the data object class can also be viewed as a dimension. For each entity from the *CVE* database, we find none or one entry from the *NVD* database, and none or one or multiple entries from the correlated Vendor database, and also entries from the third-party analyzer database (not shown in Figure 6.2). Note that an entity from the *VendorEntity* can also be connected to those data object classes like *CVSSObject*, which is not illustrated in the same figure to ensure readability. Furthermore, each vulnerability entity from the *NVD* database can be mapped to eight dimensions to support vulnerability assessment. For instance, the dimension of *CWEObject* contains attributes like weakness description, PoC exploit example, exploit likelihood, possible consequence and related attack(s). As vulnerability data sources are either unstructured (e.g., *CVE*) or semi-structured (e.g., *NVD*), this thesis chooses NoSQL database to store vulnerability related information.

### 6.1.2 DATA CORRELATION USING COMMON TAGS

Data correlation using the combination of data feeds delivers more insightful indicators compared to individual feeds. Such data integration procedure requires common tags or entities, such as *CVE ID*, affected product name and affected version number, across the ingested data feeds. This section presents a fusion method using *CVE ID* as a common tag for reports query, followed by correlation approach discussions using *CWE-ID*s, *CAPEC-ID* and *ATT&CK-ID* as common tags for weakness and attack features. Examples of the correlation processes are illustrated in Figure 6.3 and Figure 6.4.

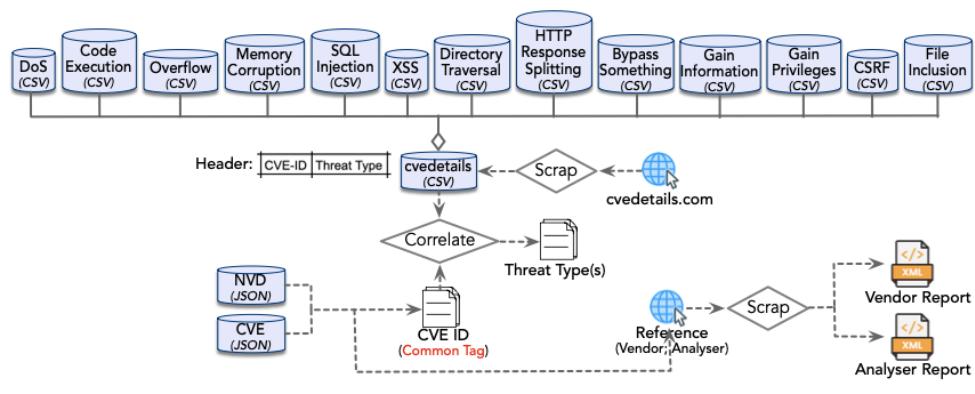


Figure 6.3: Data correlation using *CVE-ID* as the common tag

Normally, *CVE* and *NVD* reports contain references to the affected vendors and third-party analysts. These references contain URLs that can be fetched to scrap information from vendors' and security analysts' websites. With the vulnerability *CVE-ID*, URL links for additional third-party analysts are also accessible. Furthermore, scraped contents are exported in CSV or XML formats for further cleaning and analysis. Figure 6.3 presents an example that uses *CVE-ID*s to crawl the specific link within the *cvedetails.com* domain, and then scrap vulnerability reports to fetch threat category information.

Simultaneously, *CWE-ID*s are fetched from the vulnerability reports and are used as tags

to retrieve the *CWE* version 4.6 document for the matching attributes for these *CWE-ID(s)*, particularly names, descriptions and correlated *CAPEC-ID(s)*. These fetched *CAPEC-ID(s)* are further used as tags to query the *CAPEC* version 3.6 dataset for the corresponding names, descriptions and *ATT&CK-ID(s)*. Similarly, *ATT&CK* names and descriptions are extracted from *ATT&CK* version 10 document with the list of retrieved *ATT&CK-ID(s)*, as shown in Figure 6.4. It is possible that different *CWE-ID(s)* are assigned by NVD, vendors, and other security analysts. And hence, labels are added to the features to differentiate the feature sources.

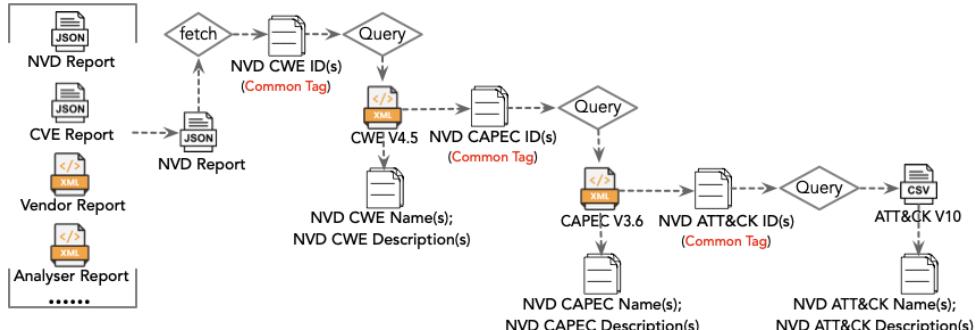


Figure 6.4: Data correlation using *CWE-ID(s)* and *CAPEC-ID(s)* as common tags

Figure 6.5 illustrates the process of correlation and retrieval of extended features for *CVE-2021-36745*, the original features of which are presented earlier in Table 2.2.

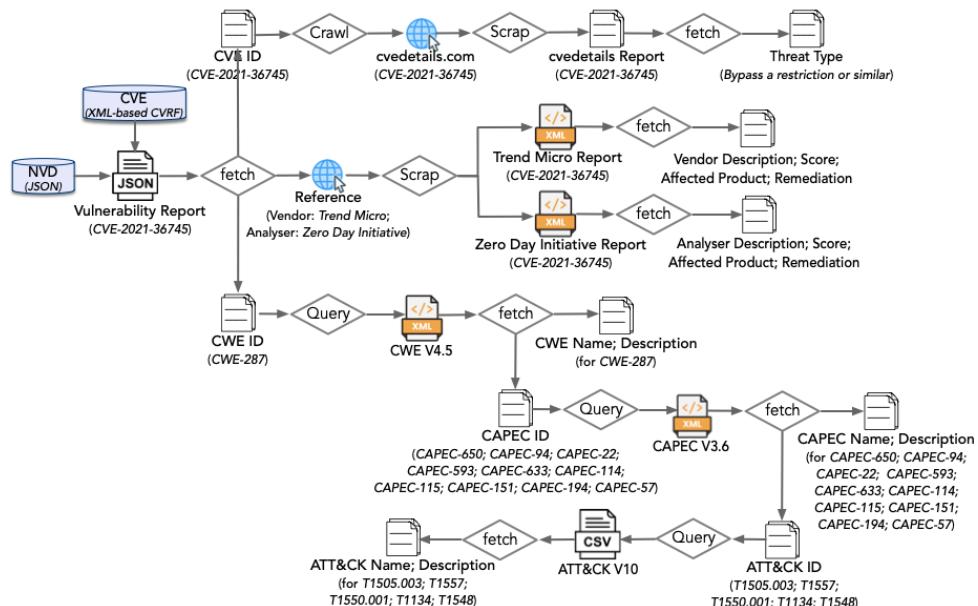


Figure 6.5: Example of data correlation for vulnerability *CVE-2021-36745*

Table 6.1 shows the additional extracted features for vulnerability instance *CVE-2021-36745*. An extended description of this instance is provided by *Zero Day Initiative* that is maintained by the affected vendor *Trend Micro*. Features regarding weakness descriptions, threats targeting this vulnerability, and potential attack techniques that may be employed to exploit this vulnerability, are consolidated to enhance the accuracy of vulnerability assessment. Besides, *CVSS* scores and *CWEs* assigned by vendors and security analysts other than *NVD* are also listed.

Table 6.1: Retrieved additional information for vulnerability *CVE-2021-36745*

Feature Name	Source	Value.
Analyst Report	<i>Zero Day Initiative</i>	This vulnerability allows remote attackers to bypass authentication on affected installations of Trend Micro ServerProtect. Authentication is not required to exploit this vulnerability. The specific flaw exists within the ServerProtect console. The issue results from the lack of proper validation prior to authentication. An attacker can leverage this vulnerability to bypass authentication on the system. <sup>1</sup>
Vendor Report Date	<i>Zero Day Initiative</i>	20210414
Public Release Date	<i>Zero Day Initiative</i>	20210926
Vendor CVSS V3 Score	Vendor: <i>Trend Micro</i>	Base score is 9.8.
Vendor CVSS V3 Vector	Vendor: <i>Trend Micro</i>	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H.
Vendor CVSS V3 Score	<i>Zero Day Initiative</i>	Base score is 9.8.
Vendor CVSS V3 Vector	<i>Zero Day Initiative</i>	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H.
Threat Category	<i>CVEDetails</i>	Bypass a restriction or similar.
<i>NVD CWE Name</i>	<i>NVD, CWE</i>	Improper Authentication.
<i>NVD Attack Category</i>	<i>NVD, CAPEC</i>	<p><i>CAPEC-650:</i> Upload a Web Shell to a Web Server.</p> <p><i>CAPEC-94:</i> Man in the Middle Attack.</p> <p><i>CAPEC-22:</i> Exploiting Trust in Client.</p> <p><i>CAPEC-593:</i> Session Hijacking.</p> <p><i>CAPEC-633:</i> Token Impersonation.</p> <p><i>CAPEC-114:</i> Authentication Abuse.</p> <p><i>CAPEC-115:</i> Authentication Bypass.</p> <p><i>CAPEC-151:</i> Identity Spoofing.</p> <p><i>CAPEC-194:</i> Fake the Source of Data.</p> <p><i>CAPEC-57:</i> Utilizing REST's Trust in the System Resource to Obtain Sensitive Data.</p>
<i>NVD Attack Technique</i>	<i>NVD, ATT&amp;CK</i>	<p><i>T1505.003:</i> Server Software Component: Web Shell</p> <p><i>T1557:</i> Man in the Middle.</p> <p><i>T1550.001:</i> Use Alternate Authentication Material: Application Access Token</p> <p><i>T1134:</i> Access Token Manipulation</p> <p><i>T1548:</i> Abuse Elevation Control Mechanism</p>
Vendor Remediation	Vendor: <i>Trend Micro</i>	<p>Critical patch is available:</p> <p>ServerProtect for EMC Celerra (SPEMC): Update to 5.8CP1577</p> <p>ServerProtect for Storage (SPFS): Update to 6.0 CP1284</p> <p>ServerProtect for Network Appliance Filers (SPNAF): Update to 5.8CP1299</p> <p>ServerProtect for Microsoft Windows / Novell Netware (SPNT): Update to 5.8CP1575</p>

We formalized the correlation process into Algorithm 1, while using a multidimensional document-oriented database schema as an example. Let  $C$  be a collection of  $n$  vulnerability documents, whereby  $C = \{D_1, \dots, D_i, \dots, D_n\}$  ( $0 < i \leq n$ ). Each vulnerability document is defined by a set of pairs between an attribute and a value, and therefore we have  $D_i = \{(Att_i^1, V_i^1), \dots, (Att_i^j, V_i^j), \dots, (Att_i^m, V_i^m)\}$  ( $0 < j \leq m$ ). The attribute value can be either atomic, or compound as a nested document composed of pairs of sub attributes and sub values. For instance, given a vulnerability document with only identity, report

and reference attribute  $D_x = \{(Att_x^{ID}, V_x^{ID}), (Att_x^{Report}, V_x^{Report}), (Att_x^{Reference}, V_x^{Reference})\}$ , we can utilise the Algorithm 1 to extract values for  $(Att_x^{Threat}, V_x^{Threat})$ ,  $(Att_x^{CWE}, V_x^{CWE})$ ,  $(Att_x^{CAPEC}, V_x^{CAPEC})$  and  $(Att_x^{ATT&CK}, V_x^{ATT&CK})$ . Value  $V_x^{Threat}$  is atomic. The other 3 values ( $V_x^{CWE}$ ,  $V_x^{CAPEC}$ , and  $V_x^{ATT&CK}$ ) are nested documents.

---

**Algorithm 1** Correlation between vulnerability instance and ATT&CK attributes
 

---

**procedure** VulnerabilityDataCorrelation( $C, D, n$ )

▷  $C$  is a collection of  $n$  vulnerability documents.

▷ Each vulnerability document  $D_i$  has identity, report and reference attributes.

$C = \{D_1, \dots, D_i, \dots, D_n\} \quad (0 < i \leq n), \quad \text{where } D_i = \{(Att_i^{ID}, V_i^{ID}), (Att_i^{Report}, V_i^{Report}), (Att_i^{Reference}, V_i^{Reference})\}$

**foreach** vulnerability instance  $D_i$  ( $i = 1, \dots, n$ ) **do**

  Retrieve vulnerability CVE ID value  $V_i^{ID}$

  Correlate  $V_i^{ID}$  with *cvedetails* database to get  $V_i^{cvedetails}$

**Set**  $V_i^{Threat} = V_i^{cvedetails}$

  Correlate  $V_i^{ID}$  with *CWE* database to get  $V_i^{CWE_{id}}$  and  $V_i^{CWE_{name}}$

**Set**  $V_i^{CWE} = \{(Att_i^{CWE_{id}}, V_i^{CWE_{id}}), (Att_i^{CWE_{name}}, V_i^{CWE_{name}})\}$

**foreach**  $cwe_{id}$  in  $V_i^{CWE_{id}}$  **do**

    Correlate  $cwe_{id}$  with *CAPEC* database to get  $capec_{id}$  and  $capec_{name}$

    Add  $capec_{id}$  to  $V_i^{CAPEC_{id}}$

    Add  $capec_{name}$  to  $V_i^{CAPEC_{name}}$

**Set**  $V_i^{CAPEC} = \{(Att_i^{CAPEC_{id}}, V_i^{CAPEC_{id}}), (Att_i^{CAPEC_{name}}, V_i^{CAPEC_{name}})\}$

**foreach**  $capec_{id}$  in  $V_i^{CAPEC_{id}}$  **do**

    Correlate  $capec_{id}$  with *ATT&CK* database to get  $att\&ck_{id}$  and  $att\&ck_{name}$

    Add  $att\&ck_{id}$  to  $V_i^{ATT\&CK_{id}}$

    Add  $att\&ck_{name}$  to  $V_i^{ATT\&CK_{name}}$

**Set**  $V_i^{ATT\&CK} = \{(Att_i^{ATT\&CK_{id}}, V_i^{ATT\&CK_{id}}), (Att_i^{ATT\&CK_{name}}, V_i^{ATT\&CK_{name}})\}$

**End procedure**

---

### 6.1.3 DATA CONSOLIDATION

Various stakeholders, such as *NVD*, vendors, and third-party security analysts, may assign different descriptions, *CVSS* severity scores and *CWE* weakness categories to the same vulnerability instances. Comparing Table 2.2 and Table 6.1, for example, different vulnerability descriptions are provided by *NVD* and *Zero Day Initiative*.

The features provided by different stakeholders for the same object category are consolidated into one class or a container, to support analytics for specific vulnerability-driven security activities along the vulnerability lifecycle. Eight classes are designed for the current stage of this thesis, while allowing flexible editing of existing containers or creation of new classes, as illustrated in Figure 6.6. This data warehouse schema is built on top of the CVRF framework. The current version of the CVRF language is 1.2 (latest checked on October 30, 2021). CVRF schemata is structured with several required elements such as *Document Publisher* and *Document Tracking*, and also multiple containers such as *Product Tree* and *Vulnerability* that further contains elements like *Threats*, *CWE*, and *CVSS Score Sets*.

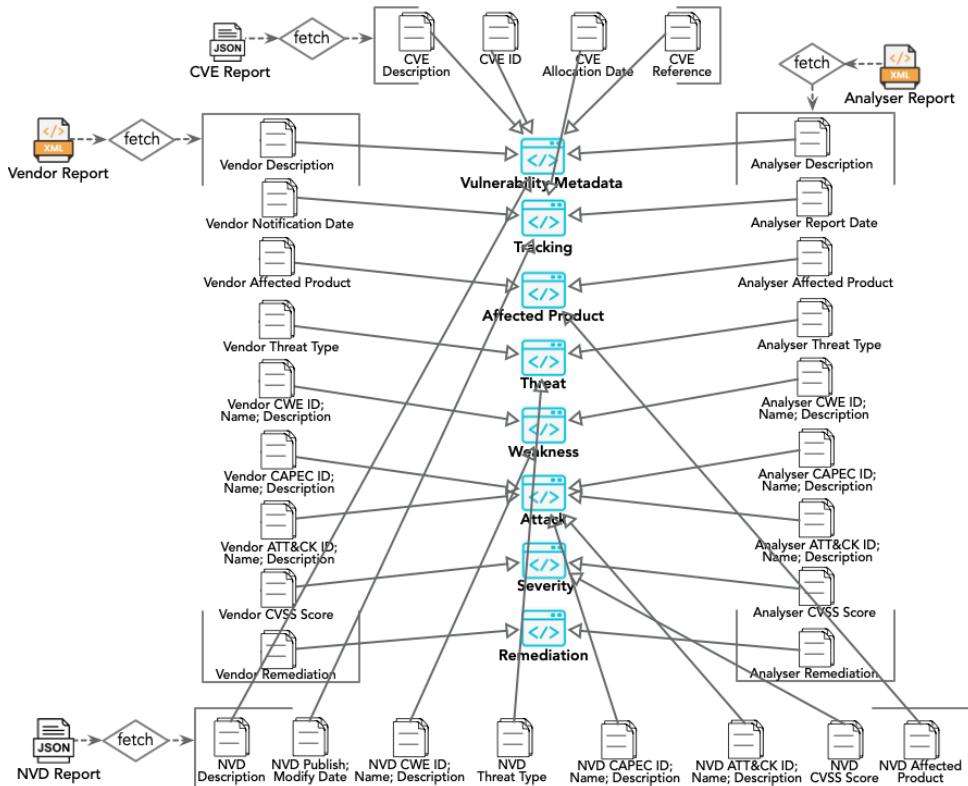


Figure 6.6: Data consolidation structure

More specifically, the *Vulnerability Metadata* class wraps up basic information like *CVE-ID*, *CVE* report description, references, and reports provided by vendors and other security analysts. *Tracking* class stores time-related information like publication dates in various data sources, reflecting the vulnerability in the lifecycle. *Affected Product* class takes in security-related software flaws, misconfigurations, and other vulnerable configuration information. *Threat* class gathers threat types that the vulnerability may be exploited, which are one or more categorical threat types in *cvedetails.com*. *Weakness* class collects information concerning weakness patterns in terms of *CWE* terminology in the investigated vulnerability. Similarly, *Attack* class aligns the vulnerability to the attack patterns using *CAPEC* identifiers and related tactics, techniques and possible implementation procedures provided by *ATT&CK*. *Severity* class captures vulnerability severity scores and matching vectors under the *CVSS V2* and *V3* mechanisms. And lastly, *Remediation* class provides mitigation suggestions provided by vendors and third-party security analysts.

An example of our proposed database structure export is shown in *Appendix III* for vulnerability instance *CVE-2021-36745*. This vulnerability example documented in JSON illustrates how we categorize cybersecurity information into our database schema.

We utilize normalized data models that use references to describe relationships or mappings between documents, as illustrated in Figure 6.7.

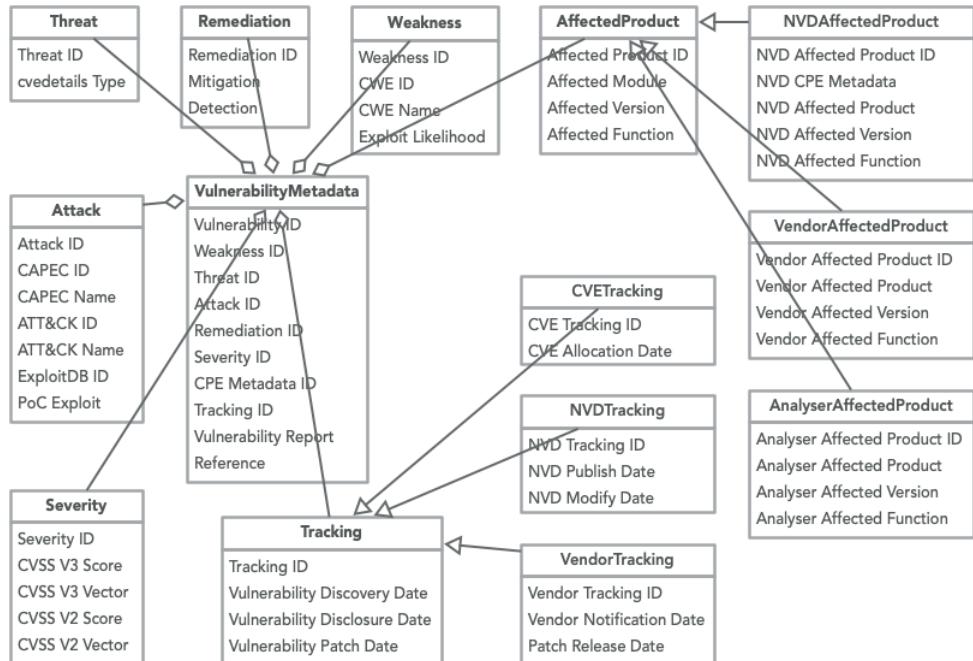


Figure 6.7: Common data model: data warehouse conceptual schema  
(Note that this is a simplified version to ensure readability)

#### 6.1.4 QUERY TAG GENERATOR

Another problem of using multiple cybersecurity data sources is synonyms as discussed earlier in Chapter 5. When query *NVD* database using tags based on vendors' product terminology, one may get incorrect vulnerability instances. It occurs when several data sources describe the same (or overlapping) set of objects and assign them different identifiers (or in this case unique attributes such as product names). Such synonym issues raise challenges when we need to query cybersecurity repositories while only holding vendor defined product names. To solve this issue, we defined a query-keywords generator that automatically suggests query tags based on similarity between the investigated product and CPE metadata, as illustrated in Figure 6.8.

The query generating process is composed of three major steps:

- In the first step, we parse the CPE metadata and extract vendor (e.g., “*microsoft*”), product (e.g., “*codeql*”) and version (e.g., “*1.0.0*”) information from these metadata. We canonize these extracted items into one entity as “*Vendor Product Version*” (e.g., “*microsoft codeql 1.0.0*”), which results in a dictionary of 816875 such entities. We further generate a dictionary using a shortened CPE metadata as key, and then using our generated entity as value, which is stored in *ProductDB* shown in Figure 6.8.

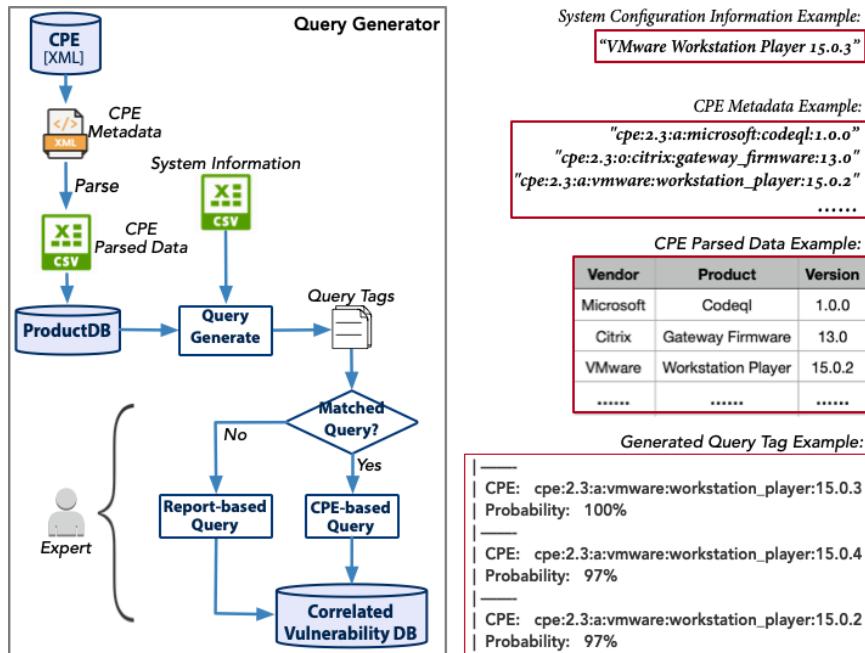


Figure 6.8: Generate query tags

- In the second step, we generate a list of query tags ranked by matching similarities. To do so, we canonize system configuration information that is usually a result of system scan into “*Vendor Product Version*”(e.g., “*vmware woskstation player 15.0.3*”) value pairs. We use the vendor information (e.g., “*vmware*”) to filter out entities in *ProductDB* from other vendors, then we generate an initial query tag list selected from the remaining entities if they partially share the tokens of software and version information (e.g. “*woskstation player 15.0.3*”). Subsequently, we measure the similarities between the system information string (e.g., “*vmware woskstation player 15.0.3*”) and strings in the initialized query tag list. By doing so, we generate a new dictionary using *CPE* metadata as key and similarity as value (e.g., ‘*cpe:2.3:a:vmware:workstation\_player:15.0.3*’: 100). We rank this query tag list from higher similarity to lower similarity, and send out the first five (can be customized to other numbers) query tags as results. We summarize this query generation process in Algorithm 2. In our approach, we compute the *Levenshtein* distance to calculate the difference between two strings, and instantiated our method by utilizing the python package *fuzzy.ratio* from *FuzzyWuzzy: Fuzzy String Matching in Python* (2021). The *Levenshtein* distance refers to the minimum number of the required single-character editing to change one string into the other (Haldar and Mukhopadhyay, 2011).
- In the last step, we allow manual check and query selection to decrease possible false positives based on other keywords that distinguish them from CI-related concepts. If we adopt one of the query tags and use *CPE*-based query, the correlated database would return vulnerability instances that share the same *CPE* metadata. If we find that all the generated query tags are not correct, we switch to report-based query and retrieve reports that contain the system configuration information string.

---

**Algorithm 2** CPE query tag generation with system configuration information

---

```

procedure VulnerabilityQueryGenerator( $D, S_1, S_2, S_3, m$ )
     $\triangleright D$  is a size  $n$  dictionary that has one key  $K_1$  and one value  $V_1$ , whereby  $K_1$  is
    a string representing CPE metadata, and  $V_1$  is a string in the format of "vendor
    product version".
     $\triangleright$  Three imported strings:  $S_1$  for vendor;  $S_2$  for product;  $S_3$  for version.
     $\triangleright m$  is an imported number.

        New query lists  $Qlist = []$ 
        Search term  $S_4 = S_2 + S_3, S_5 = S_1 + S_2 + S_3$ 
        foreach all key-value pairs  $(K_i, V_i)$  in  $D$ , ( $i=1, \dots, n$ ) do
            if  $(S_1 \text{ in } V_i) \text{ And } (S_4 \text{ in } V_i)$  then
                 $Qlist.append(K_i, V_i)$ 

        New query dictionary  $Qdict = \{\}$ 
        foreach all key-value pairs  $(K_j, V_j)$  in  $Qlist$ , ( $j=1, \dots, \text{len}(Qlist)$ ) do
            Compute strings similarity  $Similarity_j = \text{fuzzy.ratio}(S_5, V_j)$ 
             $Qdict[K_j] = \text{int}(Similarity_j)$ 

        Sort dictionary  $Qdict(K_j, Similarity_j)$  by value  $Similarity_j$ 
        Export the first  $m$  key-value pairs in  $Qdict$ 
End procedure

```

---

### 6.1.5 PROTOTYPE DATABASE DEPLOYMENT

A prototype implementation is provided, which is built on top of open-source *CVE-Search* Python API, as depicted in Figure 6.9. The data sources already correlated by *CVE-Search* and the newly correlated ones are illustrated in different colors. Besides *NVD*, *CVE* data is correlated to include the vulnerability instances that are disclosed but not yet published in *NVD*. Manufacturer websites are also correlated for standardized component names and patch updates. *cvedetails.com* website and *CWE* are correlated for threat-category information and weakness-category information separately. *SecurityFocus* and *Shodan* are also correlated to provide perspective from third-party security analysts. Using *CVE* ID as vulnerability index, a database of vulnerability reports is set up while bringing together base reports from *CVE* and cross-references from the above-mentioned repositories into a local NoSQL MongoDB system specifically designed to handle large unstructured information volumes. The proposed local MongoDB engine is kept synchronized on an hourly basis with feeds from online data repositories. Here one hour is selected as the interval by considering the vulnerability-disclosure schedule and the time needed for scanning online vulnerability resources into one local database. This localized database approach supports vulnerability analysis using queries, to retrieve structured information such as component-level vulnerability attribute values.

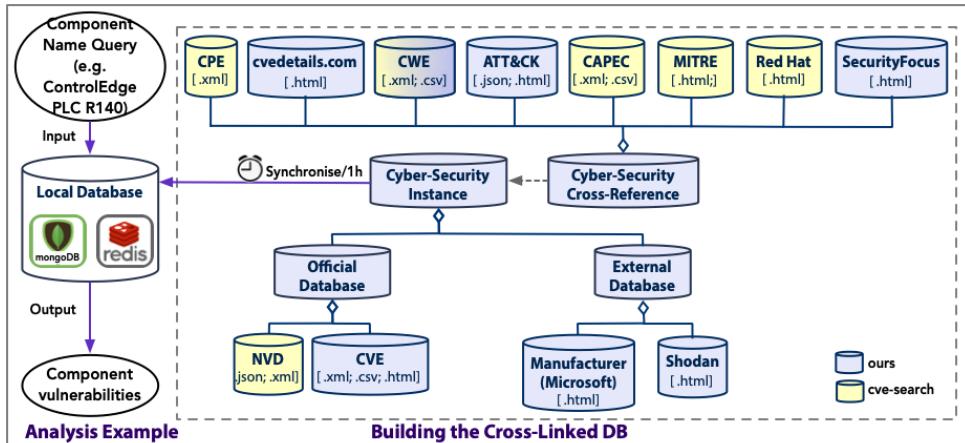


Figure 6.9: Vulnerability-database architecture and query processing  
(Figure reproduced from Jiang, Atif, and Ding (2019))

## 6.2 STUDY IV: CI VULNERABILITY TREND ANALYSIS USING CORRELATED DATABASE

The proposed vulnerability-search technique provides insights into the threat landscape in CI environments. Note that concrete information, such as the vendor, product and version information of the specific CI components, is not available at NVD. And hence, we first explored *Shodan* databases to understand some commonly used products for several prominent CI assets, namely PLC, RTU, MTU, and HMI. This study was initially conducted in 2019 July and was published in *Paper II*. Based on the original study, we renewed some results using datasets updated till November 3, 2021.

### 6.2.1 STUDY SETUP

We first investigate in *Shodan* database to extract product names, versions and vendors of industrial PLC, RTU, MTU and HMI equipments. The reason we started with *Shodan* investigation is that *Shodan* contains open ports of connected ICS devices nearly in real time. It also covers the most commonly used CPS-based CI equipments, and therefore provides actual device names, versions and vendors for our case study analysis. For example, using *PLC* as the query tag, we gather products like *Mitsubishi Q PLC*. We use these 4 lists of CI product features as input for our query generator introduced in Figure 6.8 in sub-section 6.1.4, to generate queries for our correlated database.

Vulnerability instances containing the aforementioned CI asset keywords (i.e., PLC, RTU, MTU and HMI) in their documentations are retrieved and further cross-referenced against a variety of sources. These sources include industry-standard *CPE* that is used to reveal operating systems, hardware and software information. Other cyber-security sources are also enclosed in the data fusion process, namely *CWE* that expands the information set about the vulnerability regardless the affected product instance, and *CAPEC* that provides a dictionary of known attack patterns used by adversaries to exploit the discovered vulnerabilities, as illustrated in Figure 6.10. The testing data set used as a case study are retrieved till the November 3, 2021. The *CVE* database checked on the November 3,

2021 contained 174 008 entries, *CPE* contained 357 664 entries, *CWE* entries included 1 234 elements from *CWE*, and *CAPEC* included 541 elements.

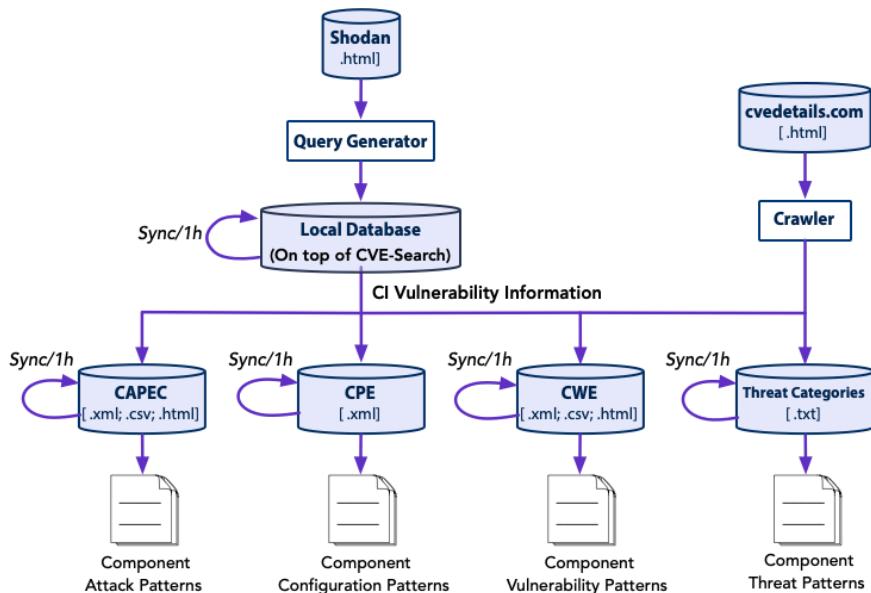


Figure 6.10: Vulnerability reports from synchronized cross-linked database  
(Figure updated from Jiang, Atif, and Ding (2019))

## 6.2.2 STUDY RESULTS

By querying our correlated database, we obtain respectively 257, 445, 107, and 258 vulnerability reports related to PLC, RTU, MTU and HMI. These retrieved 1067 CI related vulnerabilities extend till November 3, 2021. Note that some vulnerabilities appear in more than one type of CI components. One example is the vulnerability instance *CVE-2019-0708* appearing in both PLC and HMI vulnerability groups. We removed duplicated vulnerabilities and kept 767 instances in the analysis corpus when we need to assess general CI vulnerability features. Although the amounts of encountered reports for CI assets increase in recent ten years, still the total reports are limited compared to vulnerability instances for the commonly seen ones in other ICT (referring to information and communications technology) systems.

### (i) Characteristic Analysis of CI Vulnerabilities

We further analyze these identified CI vulnerabilities to get their CVSS V2 scores assigned by *NVD*. All these CI vulnerabilities are assigned CVSS V2 scores and relevant labels like V2 access vector. These assigned CVSS V2 base scores are illustrated in Figure 6.11. We conduct an investigation of the CVSS V2 labels assigned by *NVD* to our retrieved CI vulnerabilities.

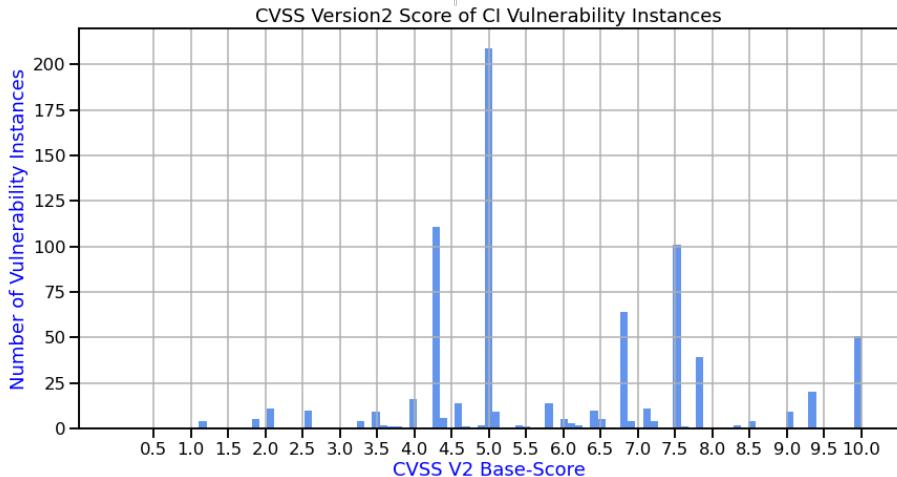


Figure 6.11: CVSS V2 base-score distribution of CI vulnerabilities (2000-2021)

Table 6.2: CI vulnerability measurement distribution of CVSS V2 base metrics

Metric	Measurement	PLC	RTU	MTU	HMI	CI	CVE
AccessVector	Network	91.44%	93.03%	90.65%	87.21 %	91.00%	83.20%
	AdjacentNetwork	1.95%	0.22%	0.93%	3.10%	1.41%	2.43%
	Local	6.61%	6.74%	8.41%	9.69%	7.59%	14.37%
AccessComplexity	Low	64.20%	64.27%	65.42%	56.59%	62.51%	58.39%
	Medium	32.30%	30.11%	28.04%	39.92%	32.80%	38.62%
	High	3.50%	5.62%	6.54%	3.49%	4.69%	2.98%
Authentication	None	90.27%	95.06%	90.65%	92.25%	92.78%	85.20%
	Single	9.73%	4.94%	9.34%	7.75%	7.22%	14.76%
	Multiple	0%	0%	0%	0%	0%	0.04%
ConfidentialityImpact	None	50.58%	46.29%	42.05%	40.31%	45.45%	32.80%
	Partial	43.58%	42.47%	50.47%	42.64%	43.58%	49.05%
	Complete	5.84%	11.24%	7.48%	17.06%	10.97%	18.15%
IntegrityImpact	None	49.42%	48.76%	54.21%	43.02%	48.08%	29.76%
	Partial	44.74%	40.90%	38.32%	41.09%	41.61%	52.64%
	Complete	5.84%	10.34%	7.48%	15.89%	10.31%	17.60%
AvailabilityImpact	None	28.02%	30.34%	37.38%	33.33%	31.21%	36.23%
	Partial	54.09%	55.73%	52.34%	44.57%	52.30%	42.67%
	Complete	17.89%	13.93%	10.28%	22.09%	16.49%	21.10%

Table 6.2 lists the exploitability and impact distributions of these vulnerability instances under CVSS V2 metrics. Vulnerabilities exist in the four types of CI show similar distributions in terms of access vector, access complexity, authentication, and availability impact. There is a higher probability that exploiting PLC vulnerabilities may bring lower confidentiality impact, but higher availability impact. Generally, CI vulnerabilities have higher exploitability compared to the overall reported vulnerabilities, especially in terms

of required authentication and complexity of such exploits.

We correlated retrieve CVE-IDs against the threat categorization provided in [www.cvedetails.com](http://www.cvedetails.com) to extract the threat types that a vulnerability instance may be exposed to. Note that a vulnerability could be exposed to more than one threat type. Three threat types, namely *Denial of Service*, *Overflow*, and *Execute Code*, appear to be the most typical ones that might materialize into attacks targeting CI assets, as shown in Figure 6.12. These three threat types with the highest occurrences in CI exploits are in line with the most common threat patterns in all the reported vulnerabilities shown in Figure 2.6. The threat type with the least occurrence is *File Inclusion* that is assigned to one PLC vulnerability instance. Actually, this threat-occurrence ranking also applies to HMI and PLC vulnerabilities. For instance, more than half of PLC vulnerabilities may be exposed to *Denial of Service* threats. In contrast, the top-3 threat types that RTU and MTU assets are faced with are *Denial of Service*, *Overflow*, and *Gain Information*. These patterns are valid for 553 CI vulnerabilities with assigned labels from [www.cvedetails.com](http://www.cvedetails.com). Although the extracted threat types are too general for this case, we show the usefulness of correlating multiple vulnerability repositories including [www.cvedetails.com](http://www.cvedetails.com).

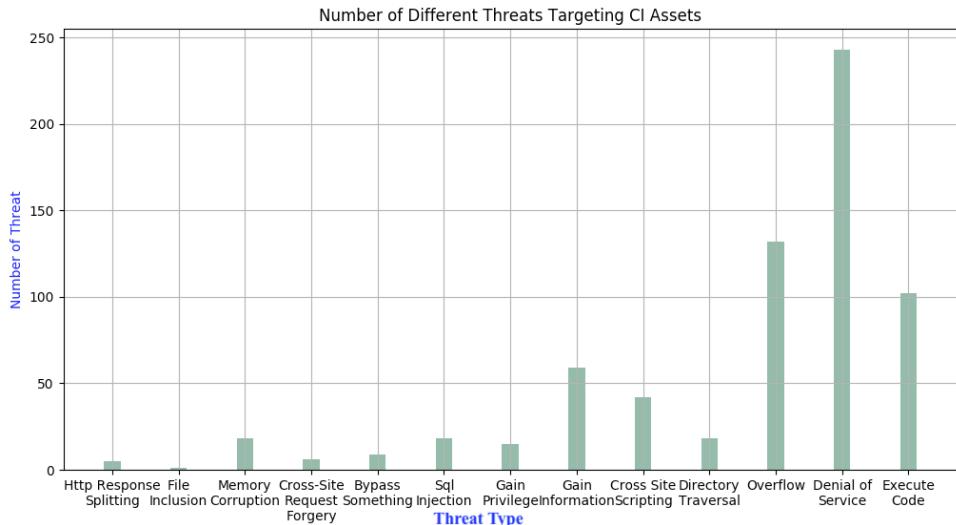


Figure 6.12: Threat types targeting CI assets (2000–2021)  
(Figure updated from Jiang, Atif, and Ding (2019))

The top-5 CWE-IDs of the retrieved CI vulnerabilities are *CWE-119* (i.e., “*Improper Restriction of Operations within the Bounds of a Memory Buffer*”), *CWE-20* (i.e., “*Improper Input Validation*”), *CWE-200* (i.e., “*Exposure of Sensitive Information to an Unauthorized Actor*”), *CWE-79* (i.e., “*Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')*”), and *CWE-787* (i.e., “*Out-of-bounds Write*”). These selected CWE-IDs have the highest occurrences in CI vulnerabilities. Apart from these common weakness patterns in CI vulnerabilities, *CWE-125* (i.e., “*Out-of-bounds Read*”) is a typical pattern for RTU, MTU and HMI vulnerabilities. Note that among all the 767 de-duplicated vulnerabilities, 248 instances have no assigned CWE categories.

Similarly, we compare the top-5 CAPEC-IDs to analyse the possible attack scenarios of our investigated four CI assets, including *CAPEC-10* (i.e., “*Buffer Overflow via Environment Variables*”), *CAPEC-14* (i.e., “*Client-side Injection-induced Buffer Overflow*”),

*CAPEC-24* (i.e., “*Filter Failure through Buffer Overflow*”), *CAPEC-42* (i.e., “*MIME Conversion*”), and *CAPEC-22* (i.e. “*Exploiting Trust in Client*”). Attack patterns are more varies for different CI assets. For instance, *CAPEC-79* (i.e., “*Using Slashes in Alternate Encoding*”) is one common attack scenario for RTU specific vulnerabilities. *CAPEC-588* (i.e., “*DOM-Based XSS*”) targets on PLC and HMI vulnerabilities more than the other two CI assets. These attack patterns are applicable for a subset of 394 vulnerabilities that are assigned both *CWE* and *CAPEC* categories.

Attack technique patterns of retrieved CI vulnerabilities are in line with *CAPEC* patterns. These attack techniques are pinpointed from *ATT&CK* standards. Note that since we correlated *ATT&CK-ID(s)* from *CAPEC-ID(s)* which are further tracked back to *CWE-ID(s)*, the presented tactic and technique patterns do not cover the CI vulnerabilities without assigned *CWE-ID(s)*. We ranked attack techniques with highest frequencies, from which the top-5 attack techniques are *T1574.007* (i.e., “*Hijack Execution Flow: Path Interception by PATH Environment Variable*”), *T1562.003* (i.e., “*Impair Defenses: Impair Command History Logging*”), *T1574.006* (i.e., “*Hijack Execution Flow: Dynamic Linker Hijacking*”), *T1134.001* (i.e., “*Access Token Manipulation: Token Impersonation/Theft*”), and *T1134.002* (i.e., “*Access Token Manipulation: Create Process with Token*”).

We also retrieve exploit patterns to identify whether the extracted CI vulnerabilities have public accessible exploits, which hints higher exploitability of those vulnerabilities. We found that 178 out of 1089 vulnerability instances (or 137 out of 767 instances after removing duplicated *CVE-ID(s)*) have PoC exploits published in *ExploitDB*. Some vulnerabilities share the same *EDB-ID(s)*. For example, *CVE-2011-4875*, *CVE-2011-4876*, *CVE-2011-4877*, *CVE-2011-4878*, and *CVE-2011-4879* were disclosed in *EDB-ID: 18166*.

## (ii) Some Examples of CI Vulnerabilities

Vulnerability in CI devices’ endpoint communication, such as using unencrypted protocols, may expose the critical data to unauthorized threat actors, and can be exploited by attacks like MiTM attacks. Vulnerability *CVE-2021-22779* illustrates such improper network segmentation weakness that has been identified in *Modicon M580* PLCs from *Schneider Electric*. Vulnerable *Modicon* PLCs employ *Schneider Electric UMAS* protocol that operates over the Modbus protocol which lacks encryption and proper authentication mechanisms. This vulnerability allows spoofing attacks to happen against the Modbus communication between the PLC controller and the *EcoStruxure* software in the engineering workstation.

Another common weakness in CPS based CI devices is improper memory access control that allow read or write operations of memory locations, which may cause out-of-bounds read and/or write. One example of such vulnerabilities is *CVE-2020-15782* (2020) that exemplifies weakness of improper operation restrictions within the bounds of a memory buffer (or *CWE-119* (2022)). This vulnerability has been identified in a list of *Siemens SIMATIC* firmware, which allows attackers with network access and download rights to a PLC to bypass existing protections in the PLC, such as PLC sandbox, and obtain read-write memory access remotely while staying undetected. A PLC sandbox refers to a protected area of memory where engineering code could run.

Outdated firmware also leaves back-doors for threat actors. Bugs in programming libraries allow more input than the assigned storage space, which may allow attackers to inject program code and overwrite original data.

### (iii) CI Vendors and Affected Products

We distinguish some specific CI manufacturers or vendor vulnerabilities reported by *Schneider Electric SE*, *Siemens AG*, and *Mitsubishi*. These instances are used as vendor data corpus to draw out relevant vulnerabilities. We discovered 29 vulnerabilities from *Schneider Electric SE* products and 39 vulnerabilities from *Siemens AG* products. We also identified 12 vulnerabilities from *Mitsubishi*.

We also observe some frequently published products that are affected by CI vulnerabilities. One typical example is *OpenSSL* that appears in 120 CPS vulnerability instances. *openSSL* (2022) is a library implementing the SSL/TLS protocol. SSL (referring to secure sockets layer) is the old name of TLS (referring to transport layer security). We also found 51 vulnerability instances related to *Simatic* PLCs and HMIs developed by *Siemens AG*.

## 6.3 STUDY V: PERFORMANCE TRADE-OFF BETWEEN THE INSTANTIATED VULNERABILITY DATABASE AND THE STATE-OF-THE-ART

This study is to compare our proposed cross-linked vulnerability database against two types of repositories: (i) public standalone vulnerability data sources, namely *CVE* and *NVD*; and (ii) state-of-the-art open-access correlation solutions such as *CVE-Search* and *Security Database*. The aim of this comparative study is to evaluate the performance trade-offs of our data correlation approach with other similar solutions, especially on usefulness and feasibility. Some other related solutions (e.g., *VERCASM-CPS* Northern et al., 2021) do not provide code sources or data sources, and could not be directly compared with our approach. Another example is *vFeed* that could not be thoroughly assessed the limited free features in its Github project page. Note that *Security Database* provides part of their features for free. We only consider these free features here, and exclude those features that require business licenses in our comparative studies. The major results of our study are listed in Table 6.3.

*CVE* and *NVD* provide reference hyperlinks to third-party cybersecurity analyzers and vendors, but do not cross-link *CVE* vulnerability instances to other sources. In comparison, the other three databases correlate vulnerability instances published in *CVE* to other vulnerability repositories such as *textitUS-CERT*, *ExploitDB*, *CERT/CC VND* and vendor security advisories such as *RedHat*, *Microsoft* and *Debian*. The free trial version of *Security Database* only partially supports vendors' source correlation. For instance, *Security Database* only updates the vendor *Microsoft* bulletin and KB till 2017.

Only *Security Database* and our approach draw attention to and provide workarounds for inconsistencies in vulnerability data sources. More specifically, *Security Database* addresses modifications in vendors' descriptions from *CVE* reports in their alerts history. In comparison, we extract vulnerability features from diverse sources and group these features into eight classes, allowing further record-level and cross-record inconsistency assessment (see details of inconsistency metrics in Chapter 5). Inconsistency root-cause analysis is also possible by combining inconsistent data attributes and vulnerability tracking information (i.e., vulnerability creation and modification timeline).

Table 6.3: Performance trade-offs between vulnerability data sources

(*Security Database* features mentioned in this Table only cover the free features that are public accessible.)

Metrics	CVE	NVD	CVE-Search	Security Database	Our Solution
Data Source	CVE	CVE	CVE	CVE	CVE
Cross-Link Data Sources	N/A	N/A	CVE, Red Hat, MITRE Maps	CVE, US-CERT, CERT/CC NVD	CVE, US-CERT, ExploitDB, RedHat Vendors (As backend)
Accessible Features Reference	CVE-ID, Report, CVSS Severity, CWE Weakness, Tracking, Reference	CVE-ID, Report, CVSS Severity, Reference, CWE Weakness, CAPEC Attack (Only Accessible through Web interface)	CVE-ID, Report, CVE Weakness, CAPEC Attack, ExploitDB exploits, CVSS Severity, Reference, Tracking	CVE-ID, Report, CAPEC and ATT&CK Attack, ExploitDB Threat, CVSS Severity, Reference	CVE-ID, Report, CAPEC and ATT&CK Attack, ExploitDB exploits, CVSS Severity, Reference, Tracking
Structured Scheme	N/A	Yes	Some (Implemented in the code but not clearly clarified)	Yes	Yes
Solution for Inconsistency	N/A	N/A	N/A	Some (By marking changes made by vendors)	Workaround (Wrapping inconsistent attributes for user to choose from)
Integrated Standards	N/A	CWE, CVSS, CPE	CWE, CAPEC, CVSS, CPE	CWE, CAPEC, CVSS, CPE, OVAL	CWE, CAPEC, CVSS, CPE, ATT&CK
Database Scheme	N/A	N/A	Document-based	N/A	Document-based
Query Support	Yes	Yes	Yes	Yes	Yes
Automation Level	Manual	Manual	Semi-Automated	Semi-Automated	Semi-Automated

All the five databases in Table 6.3 enclose vulnerability instances with assigned *CVE-ID*. This means these five databases, including our approach, do not bring additional vulnerability input than the instances disclosed in *CVE*.

*CVE-ID*, *CVE* report and references to vendors are the basic components shared by all these five databases. Besides, *NVD*, *CVE-Search*, *Security Database* and our proposed approach distinguish weakness category, affected product and severity, and tracking features by integrating cybersecurity standards *CWE*, *CPE*, and *CVSS*, separately. In addition to these four features, *CVE-Search* and our cross-linked database assign attack features to vulnerability reports through correlating *CAPEC* standards. Our database also allocates attack technique and procedure features from *ATT&CK* and threat category features from *cvedetails* to the related *CVE* reports. Yet, *CVE-Search* presents the mapping between vulnerabilities and *CAPEC* attack features only on their Web interface, but not within its database. Another issue of *CVE-Search* database is that it takes away the logical reasoning (in the form of *AND* and *OR*) within the *CPE* metadata set. More specifically, after importing *CVE* and *CPE* into the *MongoDB* based database, *CVE-Search* generates a summary report for each vulnerability. However, *CVE-Search* removes the logical correlations between *CPE* metadata, which dramatically decrease the accuracy of vulnerability retrieval. This can be exemplified by the vulnerability example *CVE-2021-1361* (2021) that affects Cisco Nexus 3000 Series Switches and Nexus 9000 Series Switches in standalone NX-OS mode that are running with Cisco NX-OS software with release version 9.3(5) or 9.3(6). The *CPE* metadata shown in *NVD* is in the format like *cpe:2.3:o:cisco:nx-os:9.35* running on/with *cpe:2.3:h:cisco:nexus\_3000*. However, this system configuration information is removed in *CVE-Search* that regards Cisco Nexus 3000 Series Switches alone as the affected product.

All the databases in Table 6.3 support text-based search queries that match the query words against vulnerability collections and retrieve the instances with these query words in their reports. Except for *CVE*, the other four databases also support *CPE* based vulnerability searching. Our approach provides supplementary support on query tags generation, which is also a workaround of inherent synonyms on product names used by *CPE* and vendors.

In summary, our proposed cross-link vulnerability database approach contributes to clear data model and correlation algorithms, and also enhances the level of automation through a query generator. While the other databases are beneficial in their own rights, our approach sets up a basis for further vulnerability attribute assessment and future simulation opportunities in various cybersecurity dimensions.

## 6.4 CONCLUSION

In this chapter, the need for, and the promises driven by correlated database management approaches to deal with unstructured and semi-structured vulnerability data are presented. We attempted to answer the research question of how to obtain and correlate vulnerability information from diverse and heterogeneous data sources. We contribute standard data models that clarifies a mapping of data sources and a data warehouse schema in UML format. We provide a correlation algorithm and multiple diagrams to illustrate attribute cross-link processes. Additionally, we designed a query generator that takes system configuration information as input and exports the best matching query tags in the format similar to *CPE* metadata. We proposed a local vulnerability database artifact, that is inherently synchronized with heterogeneous online security-related repositories using information fusion techniques to extract relevant information.

This proposed artifact is evaluated through *Study IV* that investigates vulnerability trends at both CI asset-level and component-level. *Study IV* investigates vulnerability trends of four industrial CPS devices widely used CI asset types, namely RTU, PLC, MTU and HMI. Further evaluation of retrieved vulnerabilities on actual CI assets is out of this thesis scope, yet we managed to locate information for vendors, products and version numbers from *Shodan* and vendor repositories. This case study indicates some major trends in CI vulnerabilities. It shows the potential of using the correlated vulnerability database for actual CI vulnerability assessment.

We carried out a comparative study *Study V* to compare the performance of the proposed correlated database against the baseline singular data sources and some existing data fusion solutions like *CVE-Search* and *Security Database*. This comparative study shows that our approach is beneficial compared to related methods, particularly in terms of data features, structured schema and query generation support. Meanwhile, our correlated database provides a workaround for data inconsistencies solutions at this stage. Instead of selecting one value from a set of options that represent stakeholders' various perspectives, the workaround solution is to provide a container that stores all these options, to allow the decision makers to select a preferable perspective. The correlated database cross-links multiple vulnerability data sources and brings the base for future detection and validation of data trustworthiness.

We also observed that our database allocates vulnerability features only when these features are already identified by one of the cross-linked data sources. This limitation may leave an information gap in our vulnerability database, which is usually manually assessed and assigned by cybersecurity analyzers. However, relying on individual experts' knowledge could introduce recurrent costs, subjective evaluations and contradicting outcomes. Improving the level of automation supports a timely vulnerability-analysis life-cycle. It allows mitigation within the time interval that spans the disclosure and patch of vulnerabilities. We discuss and address the limitations of information gap and manual assessment in our next chapter.

# VULNERABILITY DATA GAP ANALYSIS

Version 1.0 | Last updated: 2023-09-01

This document provides an overview of the current state of vulnerability data and identifies gaps in the available information.

The analysis is based on a review of existing datasets and literature, and aims to inform the development of new data collection and sharing mechanisms.

If you have any questions or comments, please feel free to contact us at [email address].

Thank you for your interest in this important topic.

Best regards,

The Vulnerability Data Gap Analysis Team



# CHAPTER 7

## VULNERABILITY DATA GAP ANALYSIS

The previous two chapters discuss some commonly used public vulnerability repositories and standards, where we also realized a cross-link database model that correlates these standalone data sources. In our correlated database, we categorize vulnerability-related information into eight classes that bundle together semantically related labels assigned by multiple stakeholders. Yet there are still some challenges in the direct deployment of our integrated database due to missing information (or information gap) in the original vulnerability sources. Missing information describes the fact that disclosed vulnerability instances are only partially allocated with categorical labels such as *CVSS*-metric labels that support vulnerability severity scoring and *CWE*-IDs that infer vulnerability weakness categorization. This is an obstacle that remains to be solved for a systematic and accurate vulnerability analysis and trend prediction.

This chapter focuses on answering the research question of how to bridge the missing information gap in the curated vulnerability database (as presented in Chapter 6). To answer this question, we propose two ML approaches for vulnerability assessment that infer *CVSS* severity scores and *CWE* weakness categories of reported vulnerability instances, separately. The first proposed technique also addresses compatibility issues of *CVSS* scores using a majority voting machine learning model. Vulnerability assessment and related parameters' evaluation for CI-based infrastructures use the proposed machine-learning models to provide a level of automation in vulnerability analysis. We use the correlated vulnerability database presented in Chapter 6 to get input data for ML model training and testing. For instance, we utilize the vulnerability descriptions in the class of *Vulnerability Metadata* in our database as a training input. The vulnerability entries with inconsistent scores are de-duplicated with majority voting before using it as training grounds. In doing so, ground truths for our ML-based vulnerability-severity computing algorithm are generated. We use these instances to train our ML model that is later validated with reported vulnerabilities in existing public repositories, such as *NVD* and *SecurityFocus*.

Different ML models may perform better than others for different types of cybersecurity management tasks. However, the entire group of models bundled together should hypothetically be more efficient in responding to cyber incidents, as introduced in subsection 3.2.2. Hence, this research investigates this hypothesis where multiple models are combined into a unique knowledge structure that subsumes patterns of threats, vulnerabilities, attacks, as well as their correlations. On top of the ML-based vulnerability score computing and weakness categorization approaches, we also suggest a selective ensemble model that improves the performance of ML-based algorithms based on the predefined cybersecurity task. This proposed selective ensemble meta classifier model contains a ML pipeline that processes embedded security indicators in reliable data sets, and then uses multiple ML-based components to improve predictive accuracy over individual learning algorithms. We validate this method by processing embedded security indicators across reliable data sets for severity score computation and threat categorization. The experimental analysis of actual vulnerability data sources shows some interest-

ing trade-offs of the proposed selective ensemble method, which improves the accuracy of predictions over individual ML algorithms.

This chapter covers, but is not limited to results from *Paper III*, *Paper V*, and *Paper VII*.

## 7.1 INTRODUCTION

Enterprises are increasingly facing cybersecurity flaws caused by intermittent vulnerabilities. Although recent advances in data analytic prompt dynamic data-driven vulnerability assessments whereby data contained in, and produced by complex systems include thousands of reports in different formats, still current vulnerability assessment processes are mostly conducted manually. The huge volume of data requires a high capability of information processing and analytical reasoning, which could not be satisfied considering the imprecise nature of manual vulnerability assessment. Modern security practices promote quantitative methods to provide prioritization insights and predictive analytic supported by vulnerability databases. This chapter presents multiple mechanisms based on ML and natural language processing (NLP) techniques to bridge the information gap in the vulnerability database introduced in Chapter 6. These mechanisms learn distinctive patterns from historical vulnerability data, based on which new vulnerability instances are predicted. In doing so, the proposed approach infers missing security information to enhance further cybersecurity awareness.

### 7.1.1 BRIDGING VULNERABILITY INFORMATION GAPS

*CVSS* and *CWE* are widely adopted to assess vulnerability severities and weakness exploitability across enterprises and academic research (Gawron, Cheng, and Meinel, 2017) (Johnson et al., 2016b) (Spanos, Angelis, and Toloudis, 2017). According to our static analysis in Section 5.2, around 48.08% and 5.76% vulnerability instances disclosed in *NVD* have no assigned *CVSS* V3 and V2 scores, separately. Similarly, around 27% of disclosed vulnerabilities in *NVD* have no weakness category that is needed for further assessment like PoC analysis. The missing of *CWE* categories leads to lower awareness of the root cause of disclosed vulnerabilities, and contributes to a generally longer time of vulnerability patching (Yuan et al., 2021). Actually, both *CVSS* and *CWE* exhibit some challenges when used in practice (Scarfone and Mell, 2009) (Fang et al., 2020), which are discussed next.

*CVSS*-scores are essentially influenced by individual experts, who may spend some time to rank the severity of a vulnerability since disclosed in *CVE*. This incurred time delay in evaluating vulnerabilities increases the chances of threats to materialize into actual cyber-attacks (Ruohonen, 2019). Several shortcomings need to be investigated to develop such an automatic scoring system, like inferring relevant measurements used to regulate vulnerability metrics at an appropriate scale for reported vulnerabilities. Furthermore, discrepancies among existing *CVSS* versions generate incompatible metric measurements. These issues were not fully addressed in previous research. Various organizations employ different *CVSS* versions to score vulnerability instances (Scarfone and Mell, 2009), resulting in conflicting outcomes. For example, *NVD* adopted *CVSS* version 2 in 2007 June, and then added *CVSS* version 3 scores to rate vulnerability instances reported only from 2015 December onwards. All *CVEs* have *CVSS* version 2 scores, but there are flaws in *CVSS* version 2 that *CVSS* v3 should address.

The hierarchical structure of *CWE* increases the challenge of vulnerability categorization. *CWE* version 4.6 list covers multiple views that focus on various security per-

spectives such as software development, web-based cybersecurity, and hardware weaknesses. These categorical views adopt different descriptive terms to group lower-level weaknesses. Different security communities utilize different *CWE* views when assigning weakness categories to vulnerability reports. Meanwhile, security researchers assign *CWE* weaknesses at different abstraction levels such as *Base*, *Variant*, and *Category* to vulnerability records. Due to the complex hierarchical structure of *CWE*, security experts need to navigate this structure to allocate weakness types for reported vulnerabilities. Such a process is complicated and time-consuming, resulting in weakness identification time delays.

It is desirable to define vulnerability indexes to help with a better categorization and analysis of vulnerabilities, as well as to automate the analysis process by translating natural language statements found in vulnerability reports into machine-readable formats, as a supplement to automatically generate prevalent topics in vulnerability disclosure (Hafiz and Fang, 2016). Advances in ML-based security solutions are expected to shift the burden of managing large volumes of vulnerability information away from security experts and onto some automated digital alternatives, as introduced in sub-section 3.2.1. Recently published findings report successes in applying ML techniques to improve efficiency and productivity in security activities like vulnerability alert management and related incident analysis (Edkrantz and Said, 2015) (Spanos, Angelis, and Toloudis, 2017) (Bullough et al., 2017) (Fang et al., 2020) (Sarker et al., 2020).

There is a consensus among cybersecurity experts that ML techniques does support the assessment of vulnerability data sources and hence it does alleviate some of the aforementioned cybersecurity challenges (Almukaynizi et al., 2017). In doing so, these ML approaches significantly contribute to improving the effectiveness and productivity of security activities like alert management and incident analysis. The retrieved information from vulnerability data sources supports further pattern recognition and trend analysis (Sarker et al., 2020). Using ML techniques, large amounts of such open-source vulnerability data can be analyzed (Spanos, Angelis, and Toloudis, 2017).

This thesis suggests two ML models to assign labels of CVSS metrics or *CWE* weaknesses to vulnerability instances. We use the correlated vulnerability database presented in Chapter 6 to get input data for ML model training and testing. For instance, we utilize the vulnerability descriptions in the class of *Vulnerability Metadata* of our database as a training input. The vulnerability entries with inconsistent scores are de-duplicated with majority voting before using it as training grounds. In doing so, ground truths for our ML-based vulnerability-severity computing algorithm are generated. We use these instances to train our ML model that is later validated with reported vulnerabilities in existing public repositories, such as *NVD* and *SecurityFocus*.

### 7.1.2 ENHANCING VULNERABILITY ANALYSIS

ML approaches are underutilized to streamline diverse cybersecurity tasks, such as vulnerability categorization and severity evaluation. For example, multi-label and multi-class classifications may require different ML techniques to improve prediction performances, and adopt different validation metrics (Aly, 2005) (Tsoumakas and Katakis, 2007). Since there is no single-size-that-fits-all ML-based solutions to automate cybersecurity operations, the selection of a suitable option becomes an overwhelming decision due to potential cybersecurity risks that may result from adopting the wrong ML model (Heelan, 2011). The problem addressed in this chapter is the dilemma to choose the most appropriate ML model that automatically alleviates the increasingly sophisticated threats. The objective is to assist cybersecurity professionals in keeping up with con-

stantly evolving attack tactics. The problem addressed here can be formally specified as follows:

$$\mathcal{ML} = \text{ArgMax}_{0 < i \leq N} (ML_i^D, S_p) \quad (7.1)$$

That is, given a set of vulnerability instances  $D$  and a set of  $N$  available ML baseline models  $ML_i$  ( $0 < i \leq N$ ) (e.g., a support vector machine (SVM) model), find an optimized model or ensemble of several baseline models, from an input of given classifiers  $ML_i$  trained using  $D$  labels, for analyzing multiple targeted vulnerability-analysis purposes  $S_p$ , such as categorizing vulnerability severity or threat profile.

As fact-checkers and champions of ML solutions (Veksler et al., 2018), security specialists continue to play a crucial role, devoting their intuition, ingenuity, and prior experience. This brings up the question of how to implement a ML-based security strategy that enables security management roles (e.g., operators and executives) to evaluate the risks resulting from the integration of many vulnerability sources and their combination across CI networks.

On top of the vulnerability data curation process introduced in Chapter 6 and the gap-filling ML mechanisms presented in Section 7.3 and Section 7.4 of this chapter, we further propose an ensemble approach in Section 7.5 that combines independent classifiers while taking data input from heterogeneous vulnerability sources. This approach improves the performance of vulnerability classification activities while increasing the adaptability of vulnerability analysis missions. More specifically, we suggest an optimization algorithm that selects the best ML base algorithm(s) to construct effective ensemble models for diverse cybersecurity mission targets. In addition, a variation of cross-validation is implemented to reduce the likelihood of over-fitting across classification tasks (Van der Laan, Polley, and Hubbard, 2007). The optimization algorithm analyzes all possible combination schemes for picking ML-based examples with the highest performance.

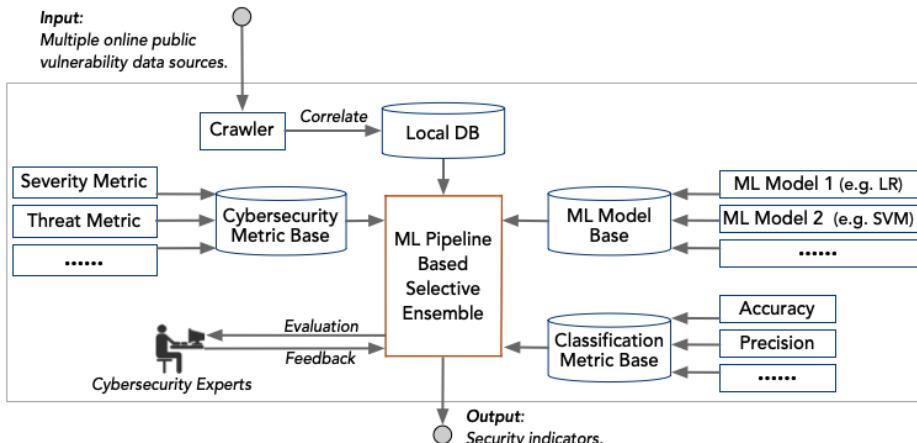


Figure 7.1: Vulnerability analysis method using ensemble ML  
(Figure reproduced from Jiang and Atif (2021))

Figure 7.1 illustrates the overall architecture of the proposed selective ensemble vulnerability analysis framework. A crawler module regularly scrapes and crawls data from

numerous online public vulnerability alert and record repositories. The collected data are then correlated and consolidated into a local database, which serves as the foundation for the subsequent ML model training and testing procedures. Particularly, a pipeline of selective ML ensemble models generates classifiers based on the entered metric sets, including vulnerability-severity metrics (e.g., confidentiality impact) and threat profile metric (e.g., DoS). The term “*pipeline*” denotes a number of ML-integrated steps chained together. Each of these ML cycles involves acquiring the data, pre-processing it, training/testing it on a chosen ML algorithm, generating some output (in the form of a prediction), and then evaluating the performance of this ML technique. In order to optimize the classification of security indicators, the proposed method also incorporates an ensemble of ML classifiers that have been trained. This ensemble model, as opposed to a single ML model, incorporates numerous algorithms under a standard knowledge framework that encompasses vulnerability patterns and their connections. Security operators participate in the validation procedure to choose and prioritize classification-performance indicators such as precision and accuracy. End-to-end optimization throughout the entire pipeline is difficult, however this chapter discusses the orchestration architecture that realizes the processes from vulnerability data curation through information-gap filling and then to pattern prediction.

Our ensemble approach combines separate classifiers that utilize information from a variety of sources, resulting in improved performance for vulnerability classification tasks and more adaptability for multiple cybersecurity missions. A form of cross-validation is utilized to reduce the likelihood of overfitting across classification jobs. Different sets of evaluation metrics are combined by our vulnerability analysis method revealed next, to select ensemble machine-learning that optimizes vulnerability severity-score computation. The classification validity for these selected ensemble machine-learning classifiers is asserted in the subsequent performance evaluation section.

## 7.2 TEXT MINING

This section introduces some text mining basics that are used in the proposed vulnerability reports assessment methods.

### 7.2.1 TEXT MINING PROCESS

Normally a text mining process includes modules for data selection, data pre-processing and cleaning, feature selection, and also ML model training, validation and optimization. In addition, some data pre-processing like tokenization and feature engineering are conducted on the textual reports and other extracted data of targeted vulnerability sets to classify new vulnerability instances, as illustrated in Figure 7.2.

#### Step 1: textual data preprocessing

Textual data preprocessing refers to the preprocessing and normalizing of returned raw textual reports, which comprises numerous typical text processing processes such as tokenization, removal of extraneous punctuation and stop words, word-stemming, and word-lemmatization (or grouping together different forms of a word into a shared item using some dictionary). Preprocessing is a crucial step in the text analysis process.

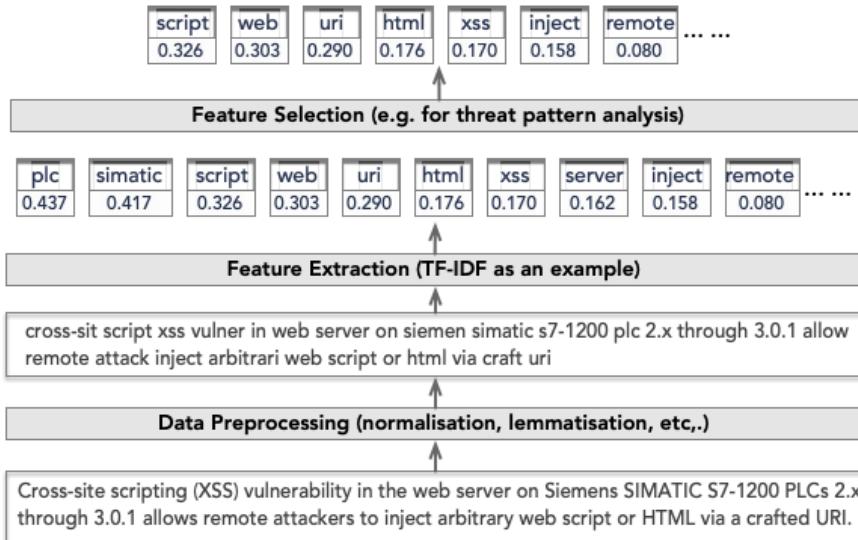


Figure 7.2: Example of data preprocessing, feature extraction and feature selection (Figure reproduced from Jiang and Atif (2021))

### Step 2: feature extraction

This step focuses on extracting relevant features from the cleaned textual reports considering the corresponding cybersecurity objective. Common textual features are categorized into indirect features and direct features. Indirect features include word counts and unique words, which are combined to form a vocabulary and term-document matrix. Each column of the matrix represents a word in the vocabulary, while each row represents a document from the dataset under investigation. In this situation, the values are the word counts (i.e., indirect feature). Direct features are those that are inherently generated from words or phrases, such as word frequency and vector distance mapping of words (e.g., with *Word2Vec*).

This thesis utilizes four ways to extract direct features, namely Count-based, Binary-based, Term Frequency-Inverse Document Frequency (TF-IDF) based, and *Word2Vec*-based feature weighting approaches (Trstenjak, Mikac, and Donko, 2014). Count-based feature extraction assigns weights to words based on the term frequency of these words in a document. One commonly used text feature extraction method named *Bag of Words* belongs to this category, which constructs a document into a vector ranked by the frequency of words. Similar to *Bag of Words*, another example of count-based feature extraction is *N-gram* approach that counts words by selecting them in groups of *N*. Binary-based weighting assigns '1' or '0' to a word depending on whether the word is presented in a document or not. TF-IDF based weighting approach assigns more weight to the words that are unique to a particular document than the words that are commonly used across all the documents (Debole and Sebastiani, 2004). TF-IDF is usually employed to extract important keywords from a document to help envisage the distinct characteristics of that document. *Word2Vec* employs clustering techniques to generate vectors for each word considering the similarities between these words.

Table 7.1: Data features, sources, types and modeling

(Table reproduced from Jiang and Atif (2021))

Feature	Feature Source	Feature Type	Feature Modeling
Vulnerability Description	CVE	Text	Word and N-Gram Character Embedding
Vulnerable Component Name	CVE, CPE	Text	Word Embedding into Vectors
Vulnerable Component Version	CVE, CPE	Number	Numeric Feature Embedding
Vulnerable Component Type	CPE	Category	One-Hot Encoding
Vulnerable Component Vendor	CVE, CPE	Text	Word Embedding into Vectors
Weakness Category ID	CWE	Number	One-Hot Encoding
Weakness Category Name	CWE	Text	Word and N-Gram Character Embedding
Weakness Category Description	CWE	Text	Word and N-Gram Character Embedding
Attack Category ID	CAPEC	Number	One-Hot Encoding
Attack Category Name	CAPEC	Text	Word and N-Gram Character Embedding
Attack Category Description	CAPEC	Text	Word and N-Gram Character Embedding
Threat Category	CVE Details	Category	One-Hot Encoding
Access Vector (Exploitability)	CVSS V2	Category	One-Hot Encoding
Access Complexity (Exploitability)	CVSS V2	Category	One-Hot Encoding
Authentication (Exploitability)	CVSS V2	Category	One-Hot Encoding
Confidentiality Impact	CVSS V2	Category	One-Hot Encoding
Integrity Impact	CVSS V2	Category	One-Hot Encoding
Availability Impact	CVSS V2	Category	One-Hot Encoding
CVSS V2 Base Score	CVSS V2	Number	Scaled to (0,10)
Attack Vector (Exploitability)	CVSS V3	Category	One-Hot Encoding
Attack Complexity (Exploitability)	CVSS V3	Category	One-Hot Encoding
Privileges Required (Exploitability)	CVSS V3	Category	One-Hot Encoding
User Interaction (Exploitability)	CVSS V3	Category	One-Hot Encoding
Confidentiality Impact	CVSS V3	Category	One-Hot Encoding
Integrity Impact	CVSS V3	Category	One-Hot Encoding
Availability Impact	CVSS V3	Category	One-Hot Encoding
CVSS V3 Base Score	CVSS V3	Number	Scaled to (0,10)
Extra Reports	Manufacture Websites; Security Blogs; etc..	Text	Word and N-Gram Character Embedding

Table 7.1 summarizes the data features, feature sources, feature types and some brief feature modeling information employed in the proposed method. Taking the vulnerability description as an example, textual features are captured and translated into vectors and embeddings representing similarities between words and n-gram characters. Another example is the numeric feature extracted from *CPE* to indicate vulnerable component type, namely hardware-, software-, and operating-system categories. We model such a numeric feature through a one-hot encoded vector to distinguish each word in the document from every other word.

### Step 3: feature selection

The purpose of this stage is to identify the most relevant features of the researched cybersecurity issue. Instead of evaluating all of the candidate features retrieved in the preceding phase, the candidate features are distinguished by pre-processing the origi-

nal feature space and prioritized by characteristics such as processing time. Moreover, different weights are provided to features based on the targeted vulnerability-analysis objectives. For example, *script*, *web*, *uri*, *html*, *xss* are given higher feature weights than *plc*, *simatic* when targeting threat-pattern analysis, as shown in Figure 7.2.

#### Step 4: ML model training and validation

Previous steps utilize NLP methods to convert vulnerability reports' content into a numerical format (Zhu and Dumitraş, 2016). This process simultaneously picks and trains machine learning algorithms with extracted features. The objective is to use ML approaches to classify new incoming reports based on prior observations in order to extract significant TVA patterns and create accurate forecasts of vulnerability score severity. As indicated previously, the most significant obstacle is the question of ML selection. In this method, ML models are trained, evaluated, and validated. Then, we select the models with the best performance as component models for subsequent ensemble formation. We deploy a simple model of data distribution parallelism to balance statistical efficiency and hardware efficiency, following the framework suggested in Jiang et al. (2018a). The training data (referring to historical incidents and reports of vulnerability) is partitioned. Each partition (or certain fields within the partition) is utilized to train a ML model. Meanwhile, model replicas are used to update and store parameter values. Multiple ML models are trained and evaluated offline in this manner. Once the training and testing processes are complete, the candidate model database is updated with meta-data and learned model configuration parameters such as the embedding dimensions, word-occurrence threshold.

### 7.2.2 MACHINE LEARNING ALGORITHMS

This section introduces several ML algorithms used in the experiments to demonstrate the proposed text-mining approaches. We utilize five supervised ML models, namely Logistic Regression (LR), Naive Bayes Support Vector Machine (NBSVM), Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN), Multi Layer Perceptron (MLP), as well as K-Nearest Neighbor (KNN). The detailed implementation and parameters after tuning are clarified in the experiment sections later in this chapter.

- *LR* works as a discriminative supervised-learning classifier that learns to assign a high weight to document features, and then assigns a class  $c$  to a document  $d$  by directly computing the likelihood  $P(c|d)$  (Almukaynizi et al., 2017) (Zhang, Caragea, and Ou, 2011). In more details, we utilize the LR text-mining method training using stochastic gradient descent and the cross-entropy loss, which returns a predicted class for a given document in the test set.
- *NBSVM* combines both NB and SVM models. SVM creates optimal hyperplanes, or decision boundaries, to distinctly separate observations into different classes, meaning data points falling on either side of a hyperplane can be attributed to different classes (Joachims, 2001). Methods using SVM calculate the maximum margin between the data points of different classes. Maximizing the margin distance provides some reinforcement to improve the model performance, which is usually done by acquiring support vectors where data points are closer to the hyperplanes. NB is based on Bayes theorem, and classifies text categorizations of an observation by computing the conditional probability values  $P(d|c)$  for each class  $c$ , given an observation  $d$ .
- *LSTM-RNN* works by recursively feeding the output of a previous network into the input of the current network, and take the final output after X number of recursions

(Zhou et al., 2016). RNN refers to Recurrent Neural Network and is commonly used in NLP and speech recognition. It has an internal memory which allows the neural network to perform the same function for every data input. The output is copied and sent back into the recurrent network, to be used in the current computation step. And hence, it is beneficial to use RNN for sequential data analysis. LSTM is adopted for short term memory learning and improves RNN performances in terms of potential vanishing gradient.

- *MLP* is one type of NN where all the units of the previous layer are connected with the units of the next layer (Zanaty, 2012). Between the input layer and the output layer, the hidden layer would adjust network weights through supervised learning.
- *KNN* works by comparing distances between unknown samples with distances between the closest known k-samples (Trstenjak, Mikac, and Donko, 2014).

### 7.2.3 VALIDATION METRICS FOR CLASSIFICATION ALGORITHMS

This thesis employs multiple classification and clustering algorithms to facilitate the vulnerability report analysis process. A single instance is classified into one of two categories using binary classification. One occurrence is classified into one of multiple (more than two) categories during multi-classification. In multi-label categorization, a single instance is assigned numerous labels, denoting various categories (Tsoumakas and Katakis, 2007). Typically, the assessment metrics for single-label are distinct from those for multi-label. In multi-label classification, a misclassification is no longer a hard wrong or right (Sorower, 2010). A prediction comprising a subset of the actual classes should be judged superior to one containing none of them, i.e., correctly predicting two of the three labels is preferable to predicting none. Therefore, the prediction of multi-label cases can be completely accurate, moderately accurate with varying degrees of accuracy, or completely inaccurate.

Different evaluation metrics are discussed next, with the details of the metric computing listed in Table 7.2.

#### (i) Binary-class evaluation metrics

To validate the accuracy of a categorization, we employ the confusion matrix. Compute the number of correctly classified class instances (true positives, or TP), the number of correctly classified instances that do not belong to the class (true negatives, or TN), and instances that were either incorrectly assigned to the class (false positives, or FP) or were not recognized as class instances (false negatives, or FN). On the basis of the aforementioned definitions, metrics such as accuracy, precision, recall (or sensitivity), and F-score (or F1) are calculated. Accuracy is a measure of a classifier's overall efficacy. Precision, on the other hand, refers to the proportion of true positives among all expected positives. Recall demonstrates the capability of a classifier to identify positive labels. F-score maintains a balance between precision and recall, respectively. The performance of the learning algorithm is enhanced by increasing values of accuracy, precision, recall, and F-score.

#### (ii) Multi-class single-label evaluation metrics

The multi-class single-label evaluation contains validations at both the macro- and micro-average levels. The micro-average differs from the macro-average in that it combines the weighted contributions of all classes. In contrast, the macro-average is calculated by averaging the contributions of all classes. And therefore, micro-average is preferable for

Table 7.2: Adopted ML model classification performance metrics

(Table Reproduced from Jiang and Atif (2021))

Metric	Binary-Class Evaluation	Multi-Class Single-Label Evaluation	Multi-Label Evaluation
	(TP is true positive; TN is true negative; FP is false positive; FN is false negative; $p_o$ is the observed agreement; $p_e$ is the expected agreement; C is the number of classes)		(k is the number of labels; n is the number of instances; $Y_i$ is the ground truth vector of labels for instance $x_i$ ; $Z_i$ is the predicted label vector for instance $x_i$ ; $h$ is the multi-label classifier; $I$ is the indicator function)
Accuracy	$ACC = \frac{TP+TN}{TP+TN+FP+FN}$	$maACC = \frac{\sum_{c=1}^C \frac{TP_c+TN_c}{TP_c+TN_c+FP_c+FN_c}}{C}$ $baACC = \frac{1}{C} \sum_{c=1}^C \frac{TP_c}{TP_c+FN_c}$	$mlACC = \frac{1}{n} \sum_{i=1}^n \frac{ Y_i \cap Z_i }{ Y_i \cup Z_i }$
Precision	$PRE = \frac{TP}{TP+FP}$	$maPRE = \frac{\sum_{c=1}^C \frac{TP_c}{TP_c+FP_c}}{C}$ $miPRE = \frac{\sum_{c=1}^C \frac{TP_c}{TP_c+FP_c}}{\sum_{c=1}^C (TP_c+FP_c)}$	$mlPRE = \frac{1}{n} \sum_{i=1}^n \frac{ Y_i \cap Z_i }{ Z_i }$
Recall	$REC = \frac{TP}{TP+FN}$	$maREC = \frac{\sum_{c=1}^C \frac{TP_c}{TP_c+FN_c}}{C}$ $miREC = \frac{\sum_{c=1}^C \frac{TP_c}{TP_c+FN_c}}{\sum_{c=1}^C (TP_c+FN_c)}$	$mlREC = \frac{1}{n} \sum_{i=1}^n \frac{ Y_i \cap Z_i }{ Y_i }$
F1	$F1 = \frac{2*PRE*REC}{PRE+REC}$	$maFS = \frac{2*maPRE*maREC}{maPRE+maREC}$ $miFS = \frac{2*miPRE*miREC}{miPRE+miREC}$	$mlFS = \frac{1}{n} \sum_{i=1}^n \frac{2 Y_i \cap Z_i }{ Y_i  +  Z_i }$
HammingLoss	HLS=1-ACC	(Same as Hamming Distance)	$mlHLS = \frac{1}{kn} \sum_{i=1}^n \sum_{l=1}^k [I(l \in Z_i \wedge l \notin Y_i) + I(l \notin Z_i \wedge l \in Y_i)]$
Cohen's Kappa		$\frac{p_o - p_e}{1 - p_e}$	

multi-class categorization problems with class imbalance. Macro-average evaluation of an C-class ( $C > 2$ ) classification problem is given by the average of per class evaluation, to compute macro-accuracy (maACC), macro-precision (maPRE), macro-recall (maREC), and macro F-score (maFS). Alternatively, micro-average classification is evaluated by summing up the amounts of TP, FN, TN, and FP to aggregate the contributions of all classes to compute the average metric. When using micro-average metrics, the value of micro-precision (miPRE), micro-recall (miREC) and micro F-score (miFS) are the same. Cohen's Kappa also applies to multi-class assessment, which represents the performance of the accepted classifier relative to the performance of a classifier that makes random classification assumptions. (Sokolova and Lapalme, 2009) (Aly, 2005)

#### (iii) Multi-label evaluation metrics

Multi-label classification utilizes a different set of metric formulations for accuracy, precision, recall and F-score (Sorower, 2010). For each instance, multi-label accuracy (or mlACC) is defined as the proportion of the predicted correct labels to the total number (both predicted and the ground-truth) of labels for that instance. Multi-label precision (mlPRE) is the proportion of predicted correct labels to the number of predicted labels. Multi-label recall (mlREC) is the proportion of predicted correct labels to the number of ground-truth labels. Multi-label F-score (mlFS) is the harmonic mean of mlPRE and mlREC. The exact-match-ratio (EMR) calculates the percentage of samples that correctly predicted all the labels. The hamming-loss (mlHLS) measure is also employed in our evaluation of security data classification. This metric is the fraction of incorrectly predicted labels, including the prediction error (an incorrect label is predicted) and the missing error (a relevant label is not predicted).  $I$  is the indicator function.  $Y_i$  is the ground truth.  $Z_i$  is the prediction. The lower the value of the hamming loss, the better the classifier's performance.

#### (iv) CVSS Score Evaluation Metrics

The standard mechanism *CVSS* is widely used to support quantitative vulnerability-severity assessment in both academic research(Khazaei, Ghasemzadeh, and Derhami, 2016)(Johnson et al., 2016b)(Spanos, Angelis, and Toloudis, 2017) and security-critical industrial domains (Stine et al., 2017). As part of the validation, we compare the predicted severity labels to their corresponding original severity labels and apply the accuracy assessment metrics to assess the performance of the ML model. We apply an alternative scores' distance metric to evaluate the performance of *CVSS*-characteristic classification across ML models. First, *CVSS* scores are computed by applying *CVSS* mechanism onto predicted counterparts (Scarfone and Mell, 2009). The existing score (true score, or TS) and a predicted score (or PS) are correlated as a two-dimensional vector space into one Cartesian coordinate system. We use the distance of the two vectors to measure the performance of our vulnerability score prediction system using Equation 7.2, where  $\delta$  is a threshold value. According to the documentation of *CVSS*, the acceptable deviation of *CVSS* score is a value of 0.5 in *CVSS Documentation* (2022) under the section “*A Word on CVSS v3.1 Equations and Scoring*”. We also consider that a deviation of 0.05 indicates better score prediction. Therefore, two values ( $\delta$  equals to 0.5 or 0.05) are used to evaluate the accuracy of our severity-score computing.

$$\text{ScoreAccuracy}_{macro} = \frac{\sum_{i=1}^n \{i | \frac{TS_i - PS_i}{TS_i} < \delta\}}{n}, \text{where } TS_i \text{ is a predicted } CVSS \text{ score.} \quad (7.2)$$

## 7.3 ARTIFACT II-A: DISCOVERING VULNERABILITY SEVERITY UNDER CVSS METRICS

This section describes a ML-based technique for discovering *CVSS* scores for reported instances of vulnerability for which no score has been assigned. This proposed method creates severity scores for instances of vulnerability automatically, which reduces the possibility of human error and reduces the effort required of human specialists. Following a brief description of the system architecture, each component of the vulnerability-severity computing system is introduced in details.

### 7.3.1 SYSTEM OVERVIEW

As seen in Figure 7.3, we acquire vulnerability data from open-source vulnerability sources. Simultaneously, we apply majority voting techniques (Tao et al., 2018) to deal with inconsistent scores retrieved from different *CVSS* scored reports across multiple repository sources. We employ these reconciled scores as the training ground for our proposed ML models, together with vulnerability reports. Then, we optimize the score prediction by employing a ML pipeline that classifies these occurrences based on a variety of *CVSS*-metric labels. Meanwhile, we store *CVSS* metrics from different *CVSS* versions in a knowledge base. Thus, the corresponding metric-set is retrieved through the user's query. The same goes for measurements and severity scales. In doing so, one can select any *CVSS* version to compute the corresponding score and vector for vulnerability instances. This suggested vulnerability severity computing system consists of a number of ML computational cycle-linked phases. Each integrated ML cycle generally consists of three phases. Step 1 is the collection of data. Step 2 does data pre-processing to prepare the data for training/testing processes involving an algorithm for machine learning. In Step 3, a predicted severity score is ultimately supplied. Data that has been pre-processed includes training/testing examples and classification measurements. Note that training and testing processes are not differentiated in Figure 7.3 to facilitate readability.

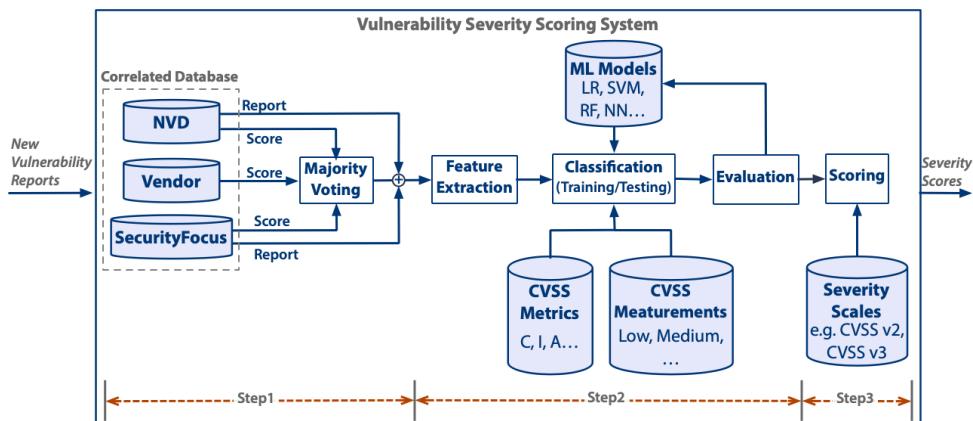


Figure 7.3: High-Level Structure of Vulnerability Severity Computing System (Figure reproduced from Jiang and Atif (2020)). More details of the correlated database shown in this figure can be found in Chapter 6

### 7.3.2 MAJORITY VOTING FOR INCONSISTENT SCORES

Relying upon *NVD* scores alone as the model training ground can bring bias in vulnerability assessment (Anwar et al., 2020) (Jo et al., 2020). This is because a small percentage of score records in *NVD* is assumed to have errors due to the manual scoring process (Scarfone and Mell, 2009). Besides statistical vulnerability patterns mined from *CVE* reports, other data sources like vendors and third-party security analysts (e.g., CERT) provide different perspectives for vulnerability scoring, as we discussed in Chapter 5 and Chapter 6. In fact, our proposed database presented in Chapter 6 correlates these diverse sources and bundle scores and other CVSS attributes assigned by these stakeholders into one class (i.e., *Severity* class).

We set up a majority voting (Tao et al., 2018) module using Python whereby the score that the majority of data sources ( $[V_1, \dots, V_d, \dots, V_D]$  where  $0 < d \leq D, D > 2$ ) in the pipeline agree on is delivered as *true score* or ground truth score. In the cases where only two score sources are found, or  $[V_1, V_2]$ , and these two scores are inconsistent, the average of these two scores are taken.

Taking the vulnerability instance *CVE-2018-7791* as an example, 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*. Nevertheless, ICS CERT assigns this vulnerability with a score of 7.7 with the vector *AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:L*. Similarly, the inconsistency comes from different measurements for attack complexity. The proposed majority voting approach suggests a final score of 9.8 as the true score. Another example of score inconsistencies is the vulnerability instance *CVE-2014-0754* which is assigned a CVSS V2 base-score 10.0 by *NVD*, VulDB and 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 inconsistency 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. We chose a final score of 10.0 as the true score using our majority voting approach.

### 7.3.3 VULNERABILITY SEVERITY COMPUTING

Vulnerability data is classified using a pipeline of ML algorithms to fill CVSS score gaps. We compare the retrieved vulnerability reports from existing vulnerability repositories against vulnerability descriptors, using text-mining approaches (Spanos, Angelis, and Toloudis, 2017). Subsequently, we classify new vulnerability reports along with CVSS-metric property groups, using a ML algorithm that is trained from a set of historical instances of reported data  $V$ . Considering  $N$  vulnerability instances from this data set,  $(v_i, Y_i)$  ( $0 < i \leq N$ ) represents a mapping between a vulnerability report  $v_i$  and a vector  $Y_i$  describing the ground truth employed by the ML algorithm.

Algorithm 3 shows the base score computation of vulnerability severity.

**Algorithm 3** Vulnerability CVSS base-score computing

---

**procedure** SeverityComputing( $\mathcal{ML}, m, M, V, V'$ )▷  $\mathcal{ML}$  is a machine learning model  $f()$ .▷  $f_{Base}()$  is the CVSS calculator, as shown in Expression 2.1▷  $[V_1, \dots, V_d, \dots, V_D]$  ( $0 < d \leq D, D > 2$ ) is a list of data sources, each of which has  $N$  vulnerability instances. Each vulnerability instance  $v_i$  ( $0 < i \leq N$ ) is assigned a list of severity scores as  $[s_{i,1}, \dots, s_{i,d}, \dots, s_{i,D}]$  and a list of CVSS vectors as  $[Y_{i,1}, \dots, Y_{i,d}, \dots, Y_{i,D}]$ .▷  $V'$  is a set of vulnerability instances  $v_p$  ( $0 < p \leq N'$ ) that have no severity score or CVSS measurements.▷  $m$  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 as maps to a value  $Y_i^{(m_j)} \in \{C_{1^{(m_j)}}, \dots, C_{k^{(m_j)}}, \dots, C_{K^{(m_j)}}\}$  ( $0 < k^{(m_j)} \leq K^{(m_j)}$ ).

$$D = |[V_1, \dots, V_d, \dots, V_D]|, N = |V_d|, N' = |V'|, M = |m|, K^{(m_j)} = |\{c_{1^{(m_j)}}, \dots, c_{k^{(m_j)}}, \dots, c_{K^{(m_j)}}\}|$$

**foreach** vulnerability instance  $v_i$  ( $i = 1, \dots, N$ ) **do**    **foreach** CVSS metric  $m_j$  ( $j = 1, \dots, M$ ) **do**

$$\text{Set } Y_i^{(m_j)} = \arg \max_{K^{(m_j)}} [\text{card}(\{c_{1^{(m_j)}}, \dots, c_{k^{(m_j)}}, \dots, c_{K^{(m_j)}}\} \| Y_{i,d}^{(m_j)})] (0 <$$

 $d \leq D$  as ground truth for CVSS measurement

$$Y_i = [Y_i^{(m_1)}, \dots, Y_i^{(m_j)}, \dots, Y_i^{(m_M)}] (j = 1, \dots, M)$$

**Set**  $s_i = f_{Base}(Y_i)$  as ground truth for severity score**foreach**  $j = 1, \dots, M$  CVSS metric  $m_j$  ( $j = 1, \dots, M$ ) **do**    Train( $\mathcal{ML}$ ) ▷  $\mathcal{ML}$  model training and testing for historical dataset

$$f^{(m_j)}(v_i) = \arg \max_{k^{(m_j)}} f_{k^{(m_j)}}^{(m_j)}(v_i)$$

**foreach** vulnerability instance  $v_p$  ( $p = 1, \dots, N'$ ) **do**    **foreach** CVSS metric  $m_j$  ( $j = 1, \dots, M$ ) **do**

$$Z_p^{(m_j)} = f^{(m_j)}(v_p) \quad \triangleright \text{Get the resulting } \mathcal{ML} \text{ predicted CVSS}$$

measurement

$$Z_p = [Z_p^{(m_1)}, \dots, Z_p^{(m_j)}, \dots, Z_p^{(M)}] (j = 1, \dots, M)$$

The resulting predicted score  $z_p = f_{Base}(Z_p)$ **End procedure**

---

CVSS metrics  $m = [m_1, \dots, m_j, \dots, m_M]$  ( $0 < j \leq M$ ), determine the  $M$  classes  $Y_i = [Y_i^{(m_1)}, \dots, Y_i^{(m_j)}, \dots, Y_i^{(m_M)}]$  where each metric class  $Y_i^{(m_j)}$  has a set of measurements  $Y_i^{(m_j)} \in \{C_{1^{(m_j)}}, \dots, C_{k^{(m_j)}}, \dots, C_{K^{(m_j)}}\}$  ( $0 < k^{(m_j)} \leq K^{(m_j)}$ ). For example, CVSS V3 is employed with the set of metrics  $m = [AV, AC, PR, UI, S, C, I, A]$  (where  $M = 8$ ), with the corresponding measurements  $Y_i = [Y_i^{(AV)}, Y_i^{(AC)}, Y_i^{(PR)}, Y_i^{(UI)}, Y_i^{(S)}, Y_i^{(C)}, Y_i^{(I)}, Y_i^{(A)}]$  such as,  $Y_i^{(AV)} \in \{N, A, L, P\}$  (where  $K^{(AV)} = 4$ ),  $Y_i^{(AC)} \in \{L, H\}$  (where  $K^{(AC)} = 2$ ),  $Y_i^{(PR)} \in \{N, L, H\}$  (where  $K^{(PR)} = 3$ ),  $Y_i^{(UI)} \in \{N, R\}$  (where  $K^{(UI)} = 2$ ),  $Y_i^{(S)} \in \{U, C\}$  (where  $K^{(S)} = 2$ ),  $Y_i^{(C)} \in \{H, L, N\}$  (where  $K^{(C)} = 3$ ),  $Y_i^{(I)} \in \{H, L, N\}$  (where  $K^{(I)} = 3$ ),  $Y_i^{(A)} \in \{H, L, N\}$  (where  $K^{(A)} = 3$ ). More information regarding the CVSS metrics can be found in sub-section 2.2.1. This definition is illustrated with the vulnerability instance introduced earlier, namely CVE-2021-37172, which can be written as (CVE-2021-37172,

$[N, L, N, N, U, N, H, N]$ .

Considering the vulnerability computing illustration shown in Figure 7.3, Lines 3-9 in Algorithm 3 represent the procedure for Step1. Lines 10-13 show the process for Step2. And finally, Lines 14-20 unfold the procedure for Step 3. The classes  $m_j$  with  $K^{(m_j)}(K^{(m_j)} > 2)$  amount of measurements , such as *AV*, is simplified into multiple binary classification problems, to differentiate between classes. Assume the employed ML model (e.g., SVM) is  $f()$ , multi-class categorization is achieved through a "one-against-all" method whereby  $f^{(m_j)}(v_i) = \arg \max_{k^{(m_j)}} f_{k^{(m_j)}}^{(m_j)}(v_i)$ .

The classification of CVSS measurements into class labels calibrates severity scores from property attributes. A *high* label of *AttackComplexity* (AC) for example, pertains to the attribute value of 0.44, and 0.77 attribute score pertains to *low* label. We use these numerical values in the CVSS calculation process.

### 7.3.4 STUDY VI: EXPERIMENT ON CVSS CATEGORIZATION EVALUATION

We utilize the contrast between severity predictions and originally labeled ones for training and testing the classification performance. Note that the experiment described in this section was conducted in 2020 and published in November of that same year. We compare our model mostly to prior publications in the same field. Accuracy, balanced accuracy, and the F1-score are utilized to evaluate this comparison. The performance implication accounts for unbalanced classes, such as *AccessVector* (*AV*) classes for example, where *Network* category has much larger sample size than *Physical* category, as depicted earlier in Figure 5.3. *AccessVector* (*AV*) classification may involve multi-class relationships, where micro-average is employed to get the mean of value across class affiliations. We employ the same strategy for all additional multi-class instances. For instance, a vulnerability occurrence is classified into one of three non-overlapping Integrity-impact measurements, i.e., high, low, or none. Binary classifiers, such as the one used for *UserInteraction* (*UI*), use a confusion matrix to determine the classification's balanced accuracy and F1-score.

We retrieve 156 040 vulnerability records corresponding to 2002 to 2020 range from *NVD* (November 3, 2021 release), and remove the reports that are marked as *REJECT* from further consideration. We set up a corpus of CVSS V2 reports by excluding reports that are not scored under CVSS V2. 148 803 vulnerability reports are subsequently filtered out, which are then correlated against trusted data sources like *ICS-CERT* (asserted by cybersecurity experts). We also use manufacturer data sources to resolve disparate scores. The proposed ML model uses these scores as ground truths for training purposes. Following the same approach, reports that are not rated under CVSS V3 are taken out to set up 75 265 instances of CVSS V3 corpus data. CVSS V3 scored reports are fewer from 2015 and earlier, with a total of 4 958 reports. The model also crawls vulnerability reports from *SecurityFocus* and map the reports to the corresponding *CVE* indexes. We add these external descriptive reports as text features for model training, together with *NVD* reports.

We implement our ML pipeline including features extraction and other data processes with the Python package *pipeline* in *Scikit-learn* library. Severity scores from different CVSS versions are thus transformed in a streamlined way. Processing *NVD* vulnerability reports' data starts from tokenization and subsequent feature extractions using *CountVectorizer* (2022) and *TfidfTransformer* (2022) utilities. Subsequently, TF-IDF

values are calculated, to generate a TF-IDF matrix from word features. *Train\_test\_split* procedure is used to randomly divide data records into training (75%) and testing (25%) datasets, following a random distribution.

Table 7.3: Performance of CVSS-metric categorization TF-IDF LR ML model

CVSS-Metric	NVD and SecurityFocus Text Features			NVD Text Features Only		
	Micro F1-Score	Balanced Accuracy	Accuracy	Micro F1-Score	Balanced Accuracy	Accuracy
V2 AccessVector(AV)	84.97 %	81.05%	95.76%	80.87%	79.53%	95.09%
V2 AccessComplexity(AC)	71.18%	64.01%	83.63%	63.68%	63.64%	84.02%
V2 Authentication(Au)	56.34%	56.21%	95.00%	57.47%	55.34%	93.92%
V2 ConfidentialityImpact(C)	81.03%	80.42%	82.98%	80.66%	79.88%	82.45%
V2 IntegrityImpact(I)	82.40%	82.04%	84.60%	82.34%	81.85%	84.43%
V2 AvailabilityImpact(A)	80.12%	80.09%	81.08%	79.44%	79.19%	80.53%
V3 AttackVector(AV)	75.92%	68.33%	93.68%	75.86%	67.93%	90.36%
V3 AttackComplexity(AC)	81.94%	75.53%	95.58%	78.78%	74.83%	95.31%
V3 PrivilegesRequired(PR)	78.79%	73.25%	90.71%	77.40%	72.50%	85.77%
V3 UserInteraction(UI)	93.45%	93.05%	94.13%	91.41%	91.00%	92.11%
V3 Scope(S)	93.65%	92.64%	97.48%	93.08%	90.66%	96.29%
V3 ConfidentialityImpact(C)	88.36%	87.74%	91.46%	84.37%	82.33%	86.67%
V3 IntegrityImpact(I)	90.58%	90.33%	92.02%	86.91%	85.79%	87.45%
V3 AvailabilityImpact(A)	75.75%	71.55%	93.01%	77.84%	70.41%	89.18%

ML classifiers predict new vulnerability reports within predicted severity patterns. Our case study uses *LR* classifier, besides a 5-fold stratified cross-validation applied to the *CVSS* training dataset to reduce overfitting occurrences. Table 7.3 illustrates *CVSS* classifier prediction performances for the testing datasets. *CVSS* V3 metric classifications reach an overall higher performance than *CVSS* V2 counterparts. However, the larger set of metrics offsets the *CVSS* V3 error rate. Table 7.3 also compares the results of using only *NVD* entries, against using both *NVD* and *SecurityFocus* entries as training features. By adding more text features, the performance of our *CVSS* scorer improves.

The outcomes assure satisfactory performances when contrasted to closely related *CVSS* classification researches from Gawron, Cheng, and Meinel (2017) as well as Yamamoto, Miyamoto, and Nakayama (2015). Gawron, Cheng, and Meinel (2017) apply Naive Bayes and Neural Networks algorithms onto *CVE* vulnerability reports published before and within 2016 to train *CVSS* version 3 classifiers. Their training dataset is adjusted to uneven the influence from data imbalance. The performance of the model proposed by Gawron, Cheng, and Meinel (2017) uses only an accuracy metric that may not adequately capture unbalanced classification instances. Nevertheless, our accuracy is higher on average. For example, the accuracy for the Attack Vector classifier is 90.36% when using only *NVD* vulnerability entries, or 93.68% when using both *NVD* and *SecurityFocus* entries. In comparison, Attack Vector classifier based on Neural Network in Gawron, Cheng, and Meinel (2017) has an accuracy of 88.9% on testing data and 80.3% on validation data. The other Attack Vector classifier based on Naive Bayes in Gawron, Cheng, and Meinel (2017) achieves an accuracy of 90.8% on testing data and 92.3% on validation data. Yamamoto, Miyamoto, and Nakayama (2015) train their *CVSS* version 2 classifiers on vulnerability instances disclosed in *NVD* from 1999 till 2014. They employed several ML algorithms, including Naive Bayes, LDA, SLDA (referring to supervised LDA), and Latent Semantic Indexing.

### 7.3.5 STUDY VII: COMPUTING SEVERITY SCORES FOR CI VULNERABILITIES

Here, we give a vulnerability analysis of four key CI components: PLC, RTU, MTU, and HMI. This study consists of two primary steps. First, we compute the CVSS V3 base scores and vectors for vulnerability occurrences that have been retrieved. Sadly, some of these vulnerabilities are not assessed or categorized until they are retrieved. Before doing further analysis, we employ our suggested approach to fill in missing or inconsistent ratings. Second, we perform an analysis to explore the statistical patterns of existing CI vulnerabilities.

We use the vulnerability datasets from *Study IV* presented in Chapter 6, whereby we extracted 257, 445, 107, and 258 vulnerability instances for PLC, RTU, MTU and HMI, separately, from our correlated database updated till November 3, 2021. We identified CVSS V2 scores for all the 1067 CI vulnerabilities, and also diagnosed multiple inconsistent scores for these instances. In contrast, 319 (71.69%) RTU vulnerabilities, 121 (47.08%) PLC vulnerabilities, 47 (43.93%) MTU vulnerabilities, and 121 (46.90%) HMI vulnerabilities are not assigned CVSS V3 scores.

We first adopt the scoring system shown in Algorithm 3 to re-compute reported vulnerability scores, as more than 57% of extracted CI vulnerability instances are not scored under the CVSS V3 mechanism. The scoring system is used to compute scores for these vulnerabilities, in order to bridge the gap of missing CVSS V3 information. We also calculate CVSS V3 scores for the vulnerabilities with inconsistent scores assigned. We design this re-computation step considering two factors, (i) CVSS V3 is only applied to vulnerabilities disclosed within and after 2015 in some data sources like NVD, and (ii) inconsistent scores are provided by multiple score sources. Subsequently, the diversity of their sub-scores is inspected to reflect CVSS V3 metric scores through property vectors evaluations. Exploitability, Scope and Impact base metric attributes for CI vulnerabilities are contrasted against actual values and illustrated in Table 7.4. CI component attributes are evaluated individually in Columns 3-6 (or Columns PLC, RTU, MTU, HMI), and averaged in Column 7 (or Column CI). Column 8 (or Column CVE) shows the overall rate of published CVE reports that have assigned CVSS V3 scores, by dividing the vulnerabilities with certain labeled measurement (e.g., Network) against all the disclosed vulnerabilities till November 3rd, 2021. By doing so, we show how the significant characteristics of CI vulnerabilities diverge when considering different vulnerability data sources.

Exploitability property attributes of CI vulnerability contrast with CVE counterparts in average, showing that a significant amount (90.48%) of attacks originate from Network-based sources, particularly for RTU vulnerabilities. There are limited occurrences of adjacent network-based attacks against CI. However, local attacks occur more frequently in CI. A large amount of CI vulnerabilities (98.72%) are prone to exploitability by malicious actors without privilege or user interaction. Change of scope is observed in 7.69% instances of CI vulnerabilities, resulting in severe consequences. A higher diversity among possible impact values is observed compared to exploitability and scope property attributes. Confidentiality and availability are more impacted than the integrity of vulnerable CI components. Nevertheless, impact of CI vulnerabilities show polarization distributions when using CVSS V3 as assessment metrics, which is the opposite when using CVSS V2 as metrics. Impact of CI vulnerabilities are mostly none or partial under CVSS V2 mechanisms. In contrast, CVSS V3 suggests that CI vulnerabilities exploitation result in either low or high compact.

CVSS V3 is used to rate CVSS severity base scores of retrieved CI vulnerability reports.

Table 7.4: CI vulnerability characteristics using CVSS version 3 base metrics

Metric	Measurement	PLC	RTU	MTU	HMI	CI	CVE
AttackVector	Network	90.27%	94.38%	93.46 %	84.11%	90.48%	74.35%
	AdjacentNetwork	1.17%	0.45%	0.93%	3.10%	1.43%	22.57%
	Local	8.17%	5.17%	5.61%	12.79%	7.96%	2.01%
	Physical	0.39%	0%	0%	0 %	1.30%	1.06%
AttackComplexity	Low	87.16%	85.39%	80.37%	91.09%	88.79%	91.21%
	High	12.84%	14.61%	19.63%	8.91%	11.21%	8.79%
PrivilegesRequired	None	85.60%	89.44%	84.11%	85.66%	88.01%	69.55%
	Low	12.84%	9.89%	15.89%	13.18%	10.69%	25.18%
	High	1.56%	0.67%	0%	1.16%	1.30%	5.28%
UserInteraction	None	84.82%	91.46%	91.59%	76.36%	84.49%	62.80%
	Required	15.18%	8.54%	8.41%	23.64%	15.51%	37.20%
ScopeChange	Unchanged	90.27%	96.18%	96.26%	87.21%	92.31%	83.64%
	Changed	9.73%	3.82%	3.74%	12.79%	7.69%	16.36%
ConfidentialityImpact	None	44.36%	41.12%	41.12%	30.62%	36.77%	22.15%
	Low	11.67%	5.62%	12.15%	13.95%	9.39%	19.10%
	High	43.97%	53.26%	46.73%	55.43%	53.85%	58.75%
IntegrityImpact	None	52.92%	50.56%	54.21%	44.96%	46.81%	31.14%
	Low	10.89%	5.62%	5.61%	13.18%	8.60%	17.20%
	High	36.19%	43.82%	40.19%	41.86%	44.59%	51.66%
AvailabilityImpact	None	29.18%	32.58%	36.45%	30.23%	29.86%	38.22%
	Low	3.11%	1.57%	3.74%	4.26%	2.22%	2.30%
	High	67.70%	65.84%	59.81%	65.50%	67.93%	61.19%

HMI, RTU and PLC vulnerability instances show high base scores at 6.5 and 8.5, respectively. MTU vulnerabilities vary within the range [4.5-8.5]. Average scores of 7.54, 8.00, 6.88 and 7.51 in CVSS V3 Base-Scores are observed respectively for PLC, RTU, MTU and HMI. Considering *CVSS V3 Documentation (2022)* qualitative scales, CI vulnerability severities are rated *medium* ([4.0-6.9]) to *high* ([7.0-8.9]).

## 7.4 ARTIFACT II-B: CATEGORIZING WEAKNESS

This section introduces some methods to deal with the complex *CWE* hierarchical structure, and an updated top-level *CWE* abstract list. We utilize this list as label input for a ML-based approach that automatically assign *CWE* categories to vulnerability reports.

### 7.4.1 HANDLING CWE TREE HIERARCHICAL STRUCTURE

Security analysts assign historical vulnerability instances various *CWE-ID(s)* with different abstract levels along the *CWE* hierarchical tree structure, which brings obstacles in weakness categorization automation. We first map all the pre-assigned *CWE-ID(s)* labels to their top-level abstracts, by tracking all the *ParentOf* relationships of the investigated *CWE-ID(s)*. A *CWE-ID* may be mapped to multiple higher-level abstracts due to various *Views*. In this research, we choose the *CWE* Research Concept View (or *CWE-1000*) (checked on November 3, 2021). We select this view for two reasons. First,

*CWE-1000* view contains all the singular weaknesses published in *CWE* and groups these weaknesses based on the abstraction of behaviors. In contrast, some other views like the Architectural Concepts view (or *CWE-1008*) only includes the architectural or design weaknesses. Secondly, *CWE-1000* view utilizes more levels of abstractions to organize weaknesses with a goal of less overlapping between these categories. In contrast, *CWE-1003* and *CWE-635* include weaknesses used by *NVD* security analysts, and are therefore useful for vulnerability report analysis, but are both organized in a shallow hierarchy.

We correlate historical vulnerability instances published in *NVD* (checked on November 3, 2021) to the *CWE-1000* Research Concept View list, by using their assigned *CWE-ID(s)* to fetch the corresponding names and child-parent relationships. The fetched parent *CWE-ID(s)* within the relationships are then used as query words to fetch the higher level *CWE-ID(s)*. We repeat this correlation process till the identified parent *CWE-ID* is in the top-level *CWE-ID(s)* entries in Table 7.5. The top-level *CWE-ID(s)* that are retrieved in the last recurrent step are the *Top CWE Abstract(s)*. For the *CWE-ID* with no parent relationship, the original *CWE-ID* is kept. Table 7.5 lists these pinpointed *CWE-ID(s)* in order of the highest occurrence to the lowest occurrence in the generated corpus. In the Column of *Abstraction*, *Category* and *Pillar* are used by *CWE* to group weaknesses, as discussed earlier in sub-section 2.2.2.

Table 7.5: Relocating the *CWE-ID*s for *NVD* vulnerability entries

(*CWE-ID*s in the 1st Column are the top categorical *CWE*s that are mapped from the original *CWE-ID*s assigned to *NVD* vulnerability instances)

<i>CWE-ID</i>	Abstraction	Covered by <i>CWE-1000</i>	Name
<i>CWE-707</i>	Pillar	Yes	Improper Neutralisation
<i>CWE-693</i>	Pillar	Yes	Protection Mechanism Failure
<i>CWE-691</i>	Pillar	Yes	Insufficient Control Flow Management
<i>CWE-710</i>	Pillar	Yes	Coding Standards Violation
<i>CWE-682</i>	Pillar	Yes	Incorrect Calculation
<i>CWE-435</i>	Pillar	Yes	Interaction Error
<i>CWE-361</i>	Category	No	Time and State
<i>CWE-264</i>	Category	No	Permissions, Privileges, and Access Controls
<i>CWE-20</i>	Category	No	Improper Input Validation
<i>CWE-399</i>	Category	No	Resource Management Errors
<i>CWE-310</i>	Category	No	Cryptographic Issues
<i>CWE-388</i>	Category	No	Error Handling
<i>CWE-189</i>	Category	No	Numeric Errors
<i>CWE-199</i>	Category	No	Information Management Errors
<i>CWE-255</i>	Category	No	Credentials Management Errors
<i>CWE-320</i>	Category	No	Key Management Errors
<i>CWE-19</i>	Category	No	Data Processing Errors
<i>CWE-254</i>	Category	No	Security Features
<i>CWE-16</i>	Category	No	Configuration
<i>CWE-371</i>	Category	No	State Issues
<i>CWE-895</i>	Category	No	Information Leak

Among all the identified *CWE-ID(s)* listed in Table 7.5, 16 entries are used by *NVD* but are not in the Research Concepts View. We manually analyze these 16 weaknesses and

map them to one of the *Pillars* at the highest level, or organize several weaknesses in the lower levels to a higher-level abstract, as presented in Table 7.6. One example is that NVD assigns both *CWE-200* and *CWE-895* to the vulnerability instance *CVE-2017-12734*. *CWE-200* implies “*Exposure of Sensitive Information to an Unauthorized Actor*”. *CWE-895* infers “*SFP Primary Cluster: Information Leak*”. In fact, the documentation of *CWE-200* covers the instruction of *CWE-895*. We then map the vulnerability instances assigned *CWE-895* to *CWE-200*, further cross-links to its parent *CWE-668*. Another example is that NVD assigns *CWE-371* to the vulnerability instance *CVE-2019-1977*. *CWE-371* has five members, namely *CWE-15*, *CWE-372*, *CWE-374*, *CWE-375* and *CWE-1265*. Among these five child members, *CWE-15* (or “*External Control of System or Configuration Setting*”) matches with the vulnerability description of *CVE-2019-1977*, which replaces *CWE-371*. There are also 4 *CWE* entries that we keep on the list, namely *CWE-19*, *CWE-264*, *CWE-16* and *CWE-254*. These 4 entries are generalized, and their assigned vulnerability instances may map to any one of the top *Pillar* entries in *CWE-1000*.

Table 7.6: Remapping of some *CWE*-IDs

<i>CWE-ID</i>	Re-Assigned <i>CWE-ID</i>	Parent <i>CWE-ID(s)</i>	Mapping Summary
<i>CWE-895</i>	<i>CWE-200</i>	<i>CWE-664</i>	<i>CWE-895 -&gt; CWE-200 -&gt; CWE-668 -&gt; CWE-664</i>
<i>CWE-19</i>	N/A	N/A	N/A
<i>CWE-264</i>	N/A	N/A	N/A
<i>CWE-255</i>	<i>CWE-522</i>	<i>CWE-664</i>	<i>CWE-255 -&gt; CWE-522 -&gt; CWE-668 -&gt; CWE-664</i>
<i>CWE-16</i>	N/A	N/A	N/A
<i>CWE-254</i>	N/A	N/A	N/A
<i>CWE-399</i>	<i>CWE-664</i>	N/A	<i>CWE-399 -&gt; CWE-664</i>
<i>CWE-310</i>	<i>CWE-710</i> <i>CWE-693</i>	N/A N/A	<i>CWE-310 -&gt; CWE-710</i> <i>CWE-310 -&gt; CWE-693</i>
<i>CWE-388</i>	<i>CWE-703</i> <i>CWE-691</i>	N/A N/A	<i>CWE-388 -&gt; CWE-703</i> <i>CWE-388 -&gt; CWE-691</i>
<i>CWE-275</i>	<i>CWE-284</i> <i>CWE-664</i>	N/A N/A	<i>CWE-275 -&gt; CWE-284</i> <i>CWE-275 -&gt; CWE-664</i>
<i>CWE-320</i>	<i>CWE-284</i>	N/A	<i>CWE-320 -&gt; CWE-284</i>
<i>CWE-189</i>	<i>CWE-19</i>	N/A	<i>CWE-189 -&gt; CWE-19</i>
<i>CWE-199</i>	<i>CWE-19</i>	N/A	<i>CWE-199 -&gt; CWE-19</i>
<i>CWE-361</i>	<i>CWE-691</i>	N/A	<i>CWE-361 -&gt; CWE-691</i>
<i>CWE-417</i>	<i>CWE-664</i>	N/A	<i>CWE-417 -&gt; CWE-664</i>
<i>CWE-371</i> (Only assigned to <i>CVE-2019-1977</i> )	<i>CWE-15</i>	<i>CWE-610</i> <i>CWE-642</i> <i>CWE-15</i>	<i>CWE-371 -&gt; CWE-15 -&gt; CWE-610 -&gt; CWE-664</i> <i>CWE-371 -&gt; CWE-15 -&gt; CWE-642 -&gt; CWE-668 -&gt; CWE-664</i> <i>CWE-371 -&gt; CWE-15 -&gt; CWE-20 -&gt; CWE-693</i>

#### 7.4.2 STUDY VIII: EXPERIMENT ON WEAKNESS CATEGORIZATION EVALUATION

We retrieved 167 532 vulnerability records from NVD with index year values ranging from 2000 to 2021 (updated till November 3, 2021), and then removed the items that match the following scenarios: (i) 5 962 reports marked as *REJECT*; (ii) 26 534 records that are assigned *NVD-CWE-Other* only; (iii) 16 308 records that are assigned *NVD-CWE-noinfo* only; and (iv) 224 records that have no *CWE* related information. The remaining 107 362 vulnerability reports construct a corpus for vulnerability analysis. We retrieved the reserved *CWE-IDs* of these vulnerability reports, and map them to the

CWE abstract list in Table 7.6. Till this step, we generated a corpus with 107 362 records. Each record has one *CVE-ID*, one vulnerability report, originally allocated *CWE-ID(s)* and the corresponding *Name* descriptions, and our assigned Abstract *CWE-ID(s)* and their descriptions.

Table 7.7: List of *CWE*-IDs for weakness categorization ML training labels

<i>CWE-ID</i>	Occurrence	Name
<i>CWE-707</i>	58756	Improper Neutralization
<i>CWE-664</i>	48193	Improper Resource Control
<i>CWE-284</i>	9425	Access Control (Authorization) Issues
<i>CWE-693</i>	8139	Protection Mechanism Failure
<i>CWE-264</i>	5322	Permissions, Privileges, and Access Controls
<i>CWE-691</i>	4533	Insufficient Control Flow Management
<i>CWE-703</i>	4364	Improper Input Validation
<i>CWE-710</i>	2404	Coding Standards Violation
<i>CWE-682</i>	1911	Incorrect Calculation
<i>CWE-19</i>	1529	Data Processing Errors
<i>CWE-254</i>	419	Security Features
<i>CWE-16</i>	267	Configuration
<i>CWE-435</i>	104	Interaction Error
<i>CWE-697</i>	41	Insufficient Comparison

The resulting dataset contains groups of vulnerability reports that are classified into different weakness types. Each class type refers to one weakness label in the format of *CWE-ID*s, i.e. one cluster of the whole data set. Table 7.8 presents one such example of the generated corpus. The report of this vulnerability instance is “*The parse\_data\_node function in bplist.c in libimobiledevice libplist 1.12 allows local users to cause a denial of service (memory allocation error) via a crafted plist file*”.

Table 7.8: Example of the generated *CWE* experiment corpus

<i>CVE-ID</i>	Report	NVD <i>CWE-ID</i>	NVD-Assigned <i>CWE</i> Name	Our <i>CWE-ID</i>	Our Assigned <i>CWE</i> Name
<i>CVE-2017-6440</i>	Report (in text)	<i>CWE-20</i> <i>CWE-190</i> <i>CWE-787</i>	Improper Input Validation; Integer Overflow or Wraparound; Out-of-bounds Write	<i>CWE-707</i> <i>CWE-682</i> <i>CWE-664</i>	Improper Neutralisation; Incorrect Calculation; Improper Control of a Resource Through its Lifetime

We randomly split the vulnerability dataset with assigned parent *CWE* categories with 75% training and 25% testing scale. We adopt *Tensorflow* and *Keras* deep learning API for text preprocessing, ML model construction and model training. These python packages are commonly used for NLP and text classification tasks. More specifically, we import *Tokenizer* and *pad\_sequences* libraries from *keras.preprocessing.text* and *keras.preprocessing.sequence*, separately. We apply *Tokenizer* to all the words in our corpus (vulnerability reports) with unique indexes, and use *pad\_sequences* to enable padding sequences of encoded data for further neural network consumption. We use a maximum of 20 000 unique words from the tokenized vulnerability report vocabulary. We utilize the pre-trained 50-dimensional *GloVe* vectors to transform each word

in a given textual data into a position in a high-dimensional space, and finally create our embedding matrix. Each line in the pre-trained *GloVe* file consists of a word and 50 numerical values. Each numerical value describes the vector of the word's position. For the words not found in the *GloVe* vectors, we randomly initialize them with the same mean and standard deviation of embeddings for the *GloVe*.

Subsequently, we utilize the built-in *keras.layers* to set up our model. We built a simple two-layer bidirectional LSTM with return-sequences set to True, a dropout-layer with probability = 0.5, and its first layer has a density of 50 classes and activation function as *relu*. Its second layer has a dense of 14 classes and uses the activation function of *sigmoid*. We defined an embedding layer that maps the words to their embedding vectors from the embedding matrix, two fully connected LSTM layers that use these vectors as input, and a *Dense* layer with *sigmoid* activation mode. The remaining parameters of the LSTM layers are shown earlier in sub-section 7.2.2. We selected the LSTM model to initialize our proposed weakness categorization approach, considering that LSTM is commonly used for NLP tasks for its eligibility of handling long sequence dependencies well. We get an accuracy of 81.5% after training for 4 epochs using the training set of 80 523 vulnerability reports.

We then use the trained model to predict the validating dataset of 26 841 vulnerability reports to evaluate our model performance. We utilize mlACC measurements from Table 7.2, namely mlACC (97.1%), mlPRE (82.2%), mlREC (82.3%), mlFS (81.5%) and mlEMR (75.5%) for model performance validation.

Our weakness categorization method using text categorization techniques investigate the missing *CWE* entries in vulnerability data sources, and also assign *CWE* categories to vulnerability instances automatically. We compare our model performances against two recent and relevant works include a NN based ThreatZoom by Aghaei, Shadid, and Al-Shaer (2020) (also see the earlier version in Aghaei and Al-Shaer (2019)) and a tree-based XGBoost classification model by Aivatoglou et al. (2021).

Table 7.9 shows that our proposed methodology predicts CWEs more or as precisely in comparison with the previous related work.

Table 7.9: Performance of *CWE* categorization LSTM ML model

(Our model performance is validated against ThreatZoom suggested by Aghaei, Shadid, and Al-Shaer (2020) and XGBoost proposed by Aivatoglou et al. (2021). Note that our model is validated using multi-label classification metrics, while the other two models are using multi-class single-label classification metrics. )

Model	Our Model	ThreatZoom	XGBoost
Accuracy	97.1%	94%	79%
Precision	82.2%	78%	73%
Recall	82.3%	75%	53%
F-Score	81.5%	77%	57%

Aghaei, Shadid, and Al-Shaer (2020) discussed both coarse-grain and fine-grain categorization of *CVEs* to *CWE* classes, whereby coarse-grain classification utilised top *CWE* nodes, and fine-grain categorization predicts the full path along the *CWE* hierarchical structure to the lower nodes. The coarse-grain performances of their adaptive hierarchical NN model tested with *NVD* dataset are listed in Table 7.9. Inspired by their works,

Aivatoglou et al. (2021) apply a set of 9 *CWE-IDs* from *MITRE* for classification purposes and used *NVD* vulnerabilities disclosed till 2021 to train three tree-based ML models, namely Random Forests, Decision Trees, and XGBoost. XGBoost achieved the best performances in their experiments. We implemented both of these works, namely a one layer NN model with 9 neurons, and an optimized gradient boosting model with 200 gradient boosted trees.

## 7.5 ARTIFACT II-C: A SELECTIVE ENSEMBLE FOR VULNERABILITY ASSESSMENT

This section presents the proposed ensemble approach that employs ML techniques to streamline cybersecurity knowledge transfer.

### 7.5.1 OVERVIEW

The proposed approach includes three steps that connect unstructured collections of data to meaningful information, which further leads to meaningful indicators and know-how at the knowledge layer level, as illustrated in Figure 7.4. Firstly, we collect heterogeneous data from public online cybersecurity repositories, periodically. Then, we classify and label curated data objects according to standard cybersecurity-related enumerations. These tagged objects are collected into a localized database with cross-linking. Chapter 6 introduces this first step in detail. Secondly, we set up an ensemble structure in the form of a pipeline while employing a set of text-classification models (Liao, Gruen, and Miller, 2020) to classify and predict relevant threat, vulnerability and attack patterns, based on data retrieved from the constructed localized database. In the final step, security experts are enabled to analyze cybersecurity indexes created by machine learning models. Their responses are fed back into the ML-based ensemble pipeline to maximize the performance of the model. This section's research focuses primarily on the second stage.

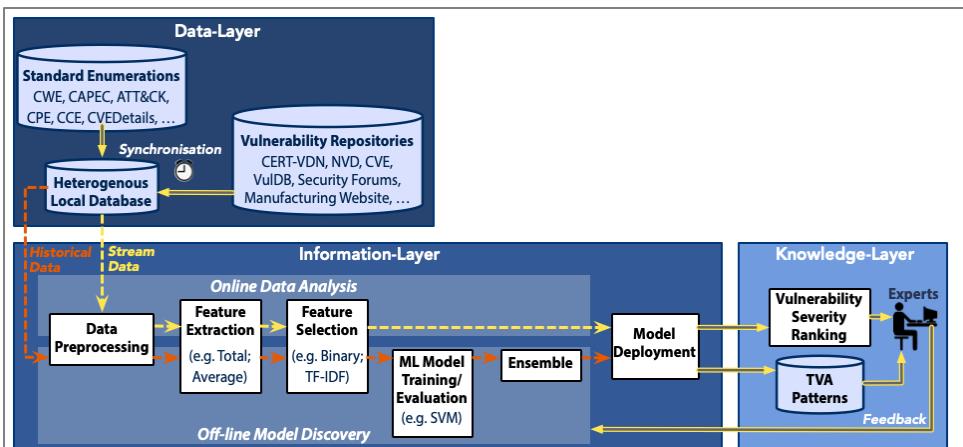


Figure 7.4: Machine learning pipeline based cybersecurity knowledge generation  
(Figure reproduced from Jiang and Atif (2021))

In the first step of the selective ensemble pipeline, we fuse diverse data streams into a local repository in a structured manner, and then combine them to trigger correlation instances. Subsequently, a ML pipeline classifies the retrieved instances to assign threat-, vulnerability-, and attack-labels accordingly. This pipeline applies text-mining (Kowsari et al., 2019), NLP, feature extraction and word embedding techniques (Zhu and Dumitras, 2016) to bridge the gap of missing information that is prevalent in textual vulnerability records, and also to predict cybersecurity trends.

## 7.5.2 CANDIDATE METRICS

This study applies different validation metrics for various classification tasks, namely binary classification tasks, multi-class classification tasks, multi-label classification tasks, as well as CVSS score evaluation metrics, as introduced before.

Multi-label accuracy (mlACC) evaluation is shown as an example to illustrate the validation process of the candidate ML models. The prediction performance  $P^{mlACC}$  of the candidate ML model is generated through Algorithm 4. Let  $D$  be a k-label dataset with  $n$  instances  $(x_i, Y_i)$  where  $Y_i \in Y = \{0, 1\}^k$  is the ground truth vector of labels for  $i$ th sample with a label-set  $|L| = k$ .  $h$  is a multi-label classifier with  $Z_i = h(x_i) = \{0, 1\}^k$  be the set of predicted label memberships for instance  $x_i$ . Given a set of  $M$  related evaluation metrics  $m_j$  ( $0 < j \leq M$ ), a performance vector  $P = [P^{m_1}, \dots, P^{m_j}, \dots, P^{m_M}]$  is generated for the candidate ML model.

---

**Algorithm 4** Performance evaluation for multi-label accuracy

---

```

procedure EvaluatePerformance(ML, m)
     $\triangleright$   $ML$  is a candidate machine learning model
     $\triangleright$   $m$  is a given machine-learning metric, which is illustrated below:
     $\triangleright$   $m \leftarrow$  Multi-Label Accuracy metric (i.e. mlACC)
     $\triangleright$  Considering a K-label ground truth dataset  $\mathcal{G}$  with total instances  $D$ , where:
         $D = |\{(Y_i, Y_j) \in \mathcal{G}, Y_j \in \{0, 1\}^K\}|$ 
     $\triangleright$   $Z$  is the resulting  $ML$  labeled data set:
         $Z = \{(Z_i, Z_j) \in ML, Z_j \in \{0, 1\}^K\}$ 

    foreach  $j = 1, \dots, D$  do
        foreach  $k = 1, \dots, K$  do
            if  $Z_{(j,k)} == Y_{(j,k)}$   $\&$   $Y_{(j,k)} == 1$  then
                 $S_j^{ACC} += 1, S_j^{Total} += 1$ 
            else
                if  $Z_{(j,k)} != Y_{(j,k)}$  then
                     $S_j^{Total} += 1$ 
             $S_j^{mlACC} = \frac{S_j^{ACC}}{S_j^{Total}}$ 
         $P^{mlACC} = \frac{1}{D} \sum_{i=1}^D P_j^{mlACC}$ 
    End procedure

```

---

### 7.5.3 MODEL SELECTION AND ENSEMBLE

The selected baseline component models build an ensemble model to aggregate the predictions of each component model and create a final estimate of the risk level associated with a single reported vulnerability. Following is additional information regarding the suggested Ensemble construction technique.

Given dataset  $D$ ,  $N$  ML models  $ML_i$  ( $0 < i \leq N$ ), and a set of  $M$  related evaluation metrics  $m_j$  ( $0 < j \leq M$ ), the following algorithmic steps construct the base ensemble of classifiers.  $N$  individual models requires  $N$  rounds of training tasks, which leads to a construction of  $\left[\binom{N}{1}, \binom{N}{2}, \binom{N}{3}, \dots, \binom{N}{N-1}, 1\right]$  amounts of ensemble models in each round. For example, in the first round, there are  $N$  ML ensemble models, of which each ensemble has only one base classifier. The second round involves  $\binom{N}{2}$  ensemble models, of which each ensemble has two base classifiers.

- Step 1: In the first round, every given individual ML model  $ML_i^{(1)}$  ( $0 < i \leq N$ ) is trained with the dataset  $D$ . The performance metric values are measured for each resulting classifier using every input metric  $m_j$  ( $0 < j \leq M$ ). At this stage, a vector of  $N$  prediction performances  $[P_1^{(1)}, \dots, P_i^{(1)}, \dots, P_N^{(1)}]$  corresponding to the first round of the framework algorithm is generated, where the prediction performance  $P_i^{(1)} = [P_{i,1}^{(1)}, \dots, P_{i,j}^{(1)}, \dots, P_{i,M}^{(1)}]$ . Therefore, the overall first round produces the following prediction performance *matrix*<sup>(1)</sup>:

$$\begin{bmatrix} P_{1,1}^{(1)} & P_{1,2}^{(1)} & \dots & P_{1,M}^{(1)} \\ P_{2,1}^{(1)} & P_{2,2}^{(1)} & \dots & P_{2,M}^{(1)} \\ \dots & \dots & \dots & \dots \\ P_{N,1}^{(1)} & P_{N,2}^{(1)} & \dots & P_{N,M}^{(1)} \end{bmatrix}$$

- Step 2: Compute rating scores  $S_i^{(1)}$  ( $0 < i \leq N$ ) for each model  $ML_i^{(1)}$ . This is done by rewarding best performing ML models  $ML_k^{(1)} = \arg \max_{ML_i^{(1)}(0 < i \leq N)} P_{i,j}^{(1)}$  under varying metrics  $m_j$  where  $j = 1 \dots M$ , with score increments.
- Step 3: Determine the best rated ML model with the highest score  $ML^{(1)} = \arg \max_{ML_i^{(1)}(0 < i \leq N)} S_i^{(1)}$  in the first algorithm round, and assert the corresponding classifier's performance vector  $P_k^{(1)} = [P_{k,1}^{(1)}, \dots, P_{k,j}^{(1)}, \dots, P_{k,M}^{(1)}]$ .
- Step 4: Repeat Step 1 to Step 3 for the remaining  $(N - 1)$  rounds which results in  $N$  best performing ensemble models from each round  $ML^{(1)}, ML^{(2)}, \dots, ML^{(N)}$ , with respectively performance vectors  $P^{(1)}, P^{(2)}, \dots, P^{(N)}$ . By the end of this iterative selection process, the following performance matrix *matrix*<sup>(final)</sup> is obtained:

$$\begin{bmatrix} P_1^{(1)} & P_2^{(1)} & \dots & P_M^{(1)} \\ P_1^{(2)} & P_2^{(2)} & \dots & P_M^{(2)} \\ \dots & \dots & \dots & \dots \\ P_1^{(N)} & P_2^{(N)} & \dots & P_M^{(N)} \end{bmatrix}$$

- Step 5: Repeat Step 2 to Step 3 to determine scores  $S^{(i)}$  ( $0 < i \leq N$ ) for each model  $ML^{(i)}$ , and determine the best performing model with the highest score  $ML = \arg \max_{ML^{(i)} (0 < i \leq N)} S^{(i)}$ , and assert the corresponding classifier's performance vector  $P_k = [P_{k,1}, \dots, P_{k,j}, \dots, P_{k,M}]$

The formal algorithm outlined through the previous steps to optimize Ensemble classifiers selection for cybersecurity analysis is depicted in Algorithm 5, where the performance evaluation for each model is depicted in Algorithm 4 explained earlier.

---

**Algorithm 5** Selective Ensemble
 

---

**procedure** SelectEnsemble( $\mathcal{ML}, m$ )

▷  $\mathcal{ML}$  is a set of individual machine learning models  $ML_i$  ( $0 < i \leq N$ ), and  $m$  is a set of related evaluation metrics  $m_j$  ( $0 < j \leq M$ )

```

 $N = |\mathcal{ML}|, M = |m|$ 
foreach Round  $r = 1, \dots, N$  do
  foreach  $i = 1, \dots, \binom{N}{r}$  do
     $ML_i^{(r)} = Ensemble(\mathcal{ML}, r)$ 
     $S_i^{(r)} = 0$            ▷ Initialize rating scores for each model at each round
    foreach  $j = 1, \dots, M$  do
      foreach  $i = 1, \dots, \binom{N}{r}$  do
         $P_{i,j}^{(r)} = EvaluatePerformance(ML_i^{(r)}, m_j)$ 
        Set  $ML_k^{(r)} = \arg \max_{ML_i^{(r)}} P_{i,j}^{(r)}, (0 < k \leq \binom{N}{r})$ 
         $S_k^{(r)} += 1$  for  $ML_k^{(r)}$  ( $0 < k \leq \binom{N}{r}$ )     ▷ Reward best performing models
      Set  $ML^{(r)} = \arg \max_{ML_i^{(r)}} S_i^{(r)}$       ▷ Assert the best ensemble model of Round  $r$ 
       $S^{(r)} = 0$ 
    foreach  $j = 1, \dots, M$  do
      Set  $ML^{(k)} = \arg \max_{ML^{(r)}} P_j^{(r)}, (0 < k \leq N)$ 
       $S^{(k)} += 1$  for  $ML^{(k)}$  ( $0 < k \leq N$ )
    Set  $ML = \arg \max_{ML^{(r)}} S^{(r)}$  as the best ensemble model
End procedure
  
```

---

It is also critical to identify an appropriate combination scheme for a selected set of individual classifiers and a given dataset (Onan, Korukoğlu, and Bulut, 2016). Soft voting ensemble and hard voting ensemble are two examples of clustering ML models for a multi-label classification task. These two ensemble mechanisms are depicted further in Algorithm 6 and Algorithm 7. Soft voting and hard voting ensemble approaches only require ML model training for the initial round, from which performance predictions for subsequent rounds can be derived. Nonetheless, the proposed ensemble paradigm can be extended to include other strategies, such as stacking methods that need numerous rounds of model training. For such applications, these models are trained offline and in parallel to best use computational resources.

---

**Algorithm 6** Soft voting ensemble scheme

---

```

procedure SoftVoting( $\mathcal{ML}, \mathcal{D}, \mathcal{K}, \mathcal{L}$ )
     $\triangleright \mathcal{ML}$  is a set of individual machine learning models  $ML_i$  ( $0 < i \leq N$ )
     $\triangleright$  K-label dataset has  $D$  instances  $(D_j, Y_j)$  ( $0 < j \leq D$ ) with ground truth  $Y_j \in Y = \{0, 1\}^k$ 
     $N = |\mathcal{ML}|, D = |(D_j, Y_j)|$ 
    foreach  $i = 1, \dots, N$  do
        Train( $ML_i$ )
         $Prob^{(i)} = \emptyset$ 
    foreach  $j = 1, \dots, D$  do
        foreach  $i = 1, \dots, N$  do
             $Prob_j^{(i)} = ML_i(D_j)$ 
         $Prob_j = \frac{1}{N} \sum_{i=1}^N Prob_j^{(i)}$ 
    End procedure

```

---



---

**Algorithm 7** Hard (majority) voting ensemble scheme

---

```

procedure HardVoting( $\mathcal{ML}, \mathcal{D}, \mathcal{K}, \mathcal{L}$ )
     $\triangleright \mathcal{ML}$  is a set of individual machine learning models  $ML_i$  ( $0 < i \leq N$ )
     $\triangleright$  K-label dataset has  $D$  instances  $(D_j, Y_j)$  ( $0 < j \leq D$ ) with ground truth  $Y_j \in Y = \{0, 1\}^k$ 
     $N = |\mathcal{ML}|, D = |d|$ 
    foreach  $i = 1, \dots, N$  do
        Train( $ML_i$ )
    foreach  $d = 1, \dots, D$  do
        foreach  $i = 1, \dots, N$  do
             $Z_j^{(i)} = ML_i(D_j)$ 
         $Z_j = \arg \max_k [card(L \| Z_j^{(i)})]$ 
    End procedure

```

---

## 7.6 STUDY IX: EXPERIMENT ON THE SELECTIVE ENSEMBLE METHOD EVALUATION

This section describes the experimental methodology's implementation and the associated analysis outcomes. Data sources are utilized to train and validate the performance of candidate ML models. The goal of the studies is to automatically infer a severity score for a new occurrence of a vulnerability and a threat type to which it is primarily vulnerable.

### 7.6.1 DATA SETS

The resulting dataset contains groups of vulnerability reports that are classified into different exploiting threat labels and CVSS labels such as access-vector types and access-

complexity levels. The goal is to concentrate attention and resources on certain acute risks deriving from threat-exploitability with varied degrees of impact-severity. Each class type refers to one label, i.e., one cluster of the whole data set. The second stage is to construct a pipeline consisting of various ML algorithms applied to distinct data class groups. Conventional methods for measuring quantitative similarity require the extraction and analysis of features. These are low-level text features that may consist of words or their compound morphology using the CHARM algorithm (Zaki and Hsiao, 2002). Both word and CHARM compound features are utilized for threat classification. At the same time, only word-features for *CVSS*-metric categorization are used. The experiments consider *CVSS* V2. These two sub-steps are conducted in tandem, to train ensembles for threat classification respectively, and to train ensembles for *CVSS*-categories.

We extract 140 818 vulnerability records with index year values ranging from 2000 to 2019 using our localized database (updated till November 30, 2020) and eliminate the reports tagged as *REJECT* that would otherwise skew the experiment results. The remaining 132 371 reports of vulnerabilities provide a corpus for vulnerability analysis.

We generate a stop-word list in the context of vulnerability reports to remove them from further consideration. We utilize *PorterStemmer* (2022) from the Natural Language Toolkit (NLTK) for word stemming implementation. Meanwhile, we employ the bi-gram functionality in *Gensim.models* to group together common bigram phrases. Some Python lemmatization tools are applied to convert a word to its root form. A TF-IDF sparse matrix is created following the dataset process using n-gram or sequence of words features. More specifically, we use CountVectorizer and TfidfTransformer utilities from *Scikit-learn* library for vectorization and TF-IDF value computation. Simultaneously, root words are sorted in descending order of TF-IDF values to extract the top-k features.

We then check these 132 371 data records in terms of threat labels, and find 39 060 or 29.5% un-labeled data. These un-labeled vulnerability reports are not categorized to any existing threat category. A vulnerability may be exposed to more than one sort of danger; therefore, a vulnerability instance may have numerous threat labels. We employ the remaining 93 311 records for threat classification training and validation as ground truth. This dataset includes 20 reports without *CVSS* V2 scores and therefore removed, which left 132 351 scored reports as ground truth for *CVSS* V2 metrics classification training and validation.

## 7.6.2 EXPERIMENT SETUP

The experiment setup includes a ML pipeline by using the existing package *pipeline* from the *Scikit-learn* library, to automate the ML workflow. This pipeline utilizes different techniques in the ML stack, such as data processing and feature extraction, according to the classification tasks and data subsets. This experiment considers five supervised ML models, namely LR, NBSVM, LSTM-RNN, MLP and KNN. These five ML models have been adopted in multi-class classification tasks towards *CVSS* prediction. For instance, Wen et al. (2015) used SVM in *CVSS* V2 classification. Le, Sabir, and Babar (2019) utilized Logistic Regression, SVM, KNN, Naive Bayes and three tree-based ensemble methods including Random Forest. While these five models are used individually in cybersecurity and text-mining applications, the proposed ensemble paradigm extends their potential capabilities together with other ML techniques in a range of vulnerability analyses. We set up two experiments on top of the ML pipeline, one for threat categorization, and the other for *CVSS* metric categorization.

We utilized several python libraries to implement the five supervised ML models. More specifically, we adopt *LogisticRegression*, *MLPClassifier*, and *KNeighborsClassifier* packages from the *Scikit-learn* library to implement LR, MLP and KNN algorithms, separately. Our model based on a NBSVM algorithm was inspired by Wang and Manning (2012) that uses the NB log-count ratios as feature values, and was implemented from scratch. We uses python library *LSTM* from *keras.layers* to build a simple two-layer bidirectional LSTM with return-sequences set to True, a dropout-layer with probability = 0.5, and its first layer has a density of 50 classes (or 50 neurons). Its second layer has a dense of 13 classes for threat categorization, or a density of 2/3/4 classes for *CVSS* categorization.

#### (i) Threat categorization

Threat categorization modeling is a multi-label classification problem. This experiment applies the “one-to-rest” strategy to solve the multi-label problem by decomposing it into multiple independent binary classification problems (one per category). We divide the retrieved 93 311 data records into a 75% (or 69 983 records) training dataset and a 25% (or 23 328 records) validation dataset, randomly. The following validation applies 3-fold stratified cross-validation using cross-val-score from the *sklearn* package on the five previously mentioned ML models for training and testing threat classification. The performances of these five learning algorithms are evaluated on an unseen validation dataset, of which the model with the best performance is chosen. In the first round of training, these independently trained models produce five files with the projected probability of various labels for each event. Instead of training multi-round ML models, this experiment uses a soft-voting technique to aggregate the projected probability of individual base ML models to calculate predictions for ensemble models.

#### (ii) CVSS categorization

Similarly, this experiment simulates an efficient vulnerability scoring system based on the *CVSS* V2 mechanism, in order to automatically assess the severity and exploitability of a vulnerability instance. We generate the experimental dataset by correlating the existing *CVSS* scores of vulnerability instances in *NVD* to other data sources, such as vendor websites and technical reports from third party reviewers, to adjust scores and better describe the actual severity of vulnerability instances. We employ these resolved scores and corresponding counterparts as the training ground for our ML models. We then divide these retrieved 132 351 data records into a 75% (or 99 263 records) training dataset and a 25% (or 33 088 records) validation dataset, randomly. Then, the experiment applies 5-fold stratified cross-validation on the training dataset to train the five afore-mentioned ML models that learn and test *CVSS* v2 categorization. Five files containing the anticipated labels of distinct *CVSS* v2 features are generated by these five trained models. The ensemble models utilize the labels with the majority of votes.

### 7.6.3 THREAT CATEGORIZATION RESULTS

Evaluation of the classification algorithms is a measurement of how far the classification systems’ predictions are from the actual class labels, tested on some unseen data.

We validate the performances of the learning algorithms on unseen validation datasets following six multi-label classification metrics introduced earlier in Section 7.4.3. The first five measurements, namely *mlACC*, *mlPRE*, *mlREC*, *mlFS* and *mlEMR*, are calculated by applying the corresponding equations. The last *hamming\_loss* metric utilises the *sklearn.metrics* package to calculate *mlHLS*.

After five training cycles, we generate 31 performance files to evaluate the performance of the pipeline algorithms and select the best model from the base Ensemble models. As further indicated in Figure 7.5, the individual models KNN and LR perform less effectively than LSTM, NBSVM, and MLP. However, the performance of the ensemble remains relatively steady when these poorer base learners are included. This is the benefit of soft-voting that considers the confidence of each base learner and not just binary choices. Except for mlHLS, the average performance of models is calculated in each round. All the other five metrics have better performance when the amount of participating base learners increase, as illustrated further in Figure 7.6.

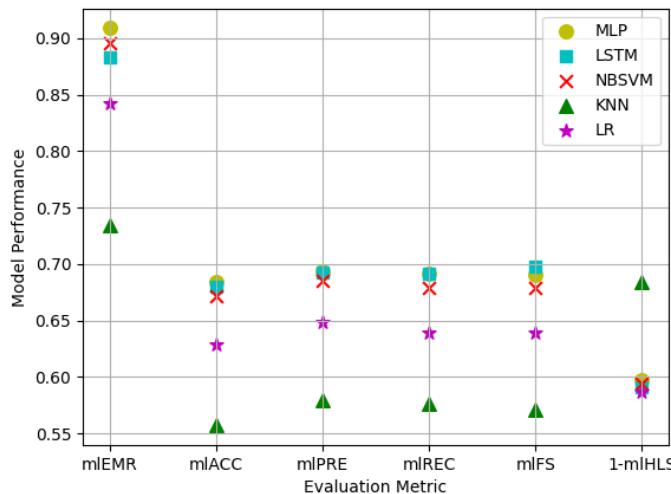


Figure 7.5: Performance of individual ML models for threat classification (Figure reproduced from Jiang and Atif (2021)). LSTM, NBSVM and MLP models have better performance than LR and KNN models.

Figure 7.7 lists best-performing models of each round. In all configurations, the pipelined ensemble model provides the best performance. In the first round, individual model MLP reveals the best performance. In the second round, the LSTM and MLP ensemble achieves the highest performance. The ensemble of LSTM, SVM, and MLP gets the best performance in the third round. The ensemble of LSTM, SVM, KNN, and MLP gets the best performance in the fourth round. And finally, the fifth round has only one ensemble model, therefore no comparative counterparts. The combination of LSTM and MLP delivers the best overall performance. But the ensemble of LSTM, NBSVM and MLP provides very close performance. Most of the predictive performances like mlACC, mlPRE, mlFS reach their peak value for the ensemble model with base learners LSTM and MLP. Interestingly, mlHLS has the best performance among individual models, suggesting that ensemble models may create greater loss in the class label bit string during prediction. To our knowledge, there is no comparable threat classification in the literature. Nonetheless, a mlEMR score of 91.63% by the ensemble of LSTM and MLP demonstrates the predictive capability of our model.

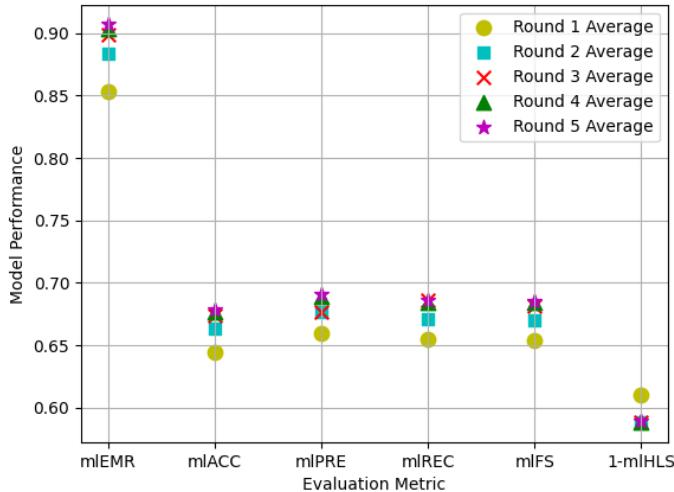


Figure 7.6: Average threat-classification model performance in each training round  
(Figure reproduced from Jiang and Atif (2021))

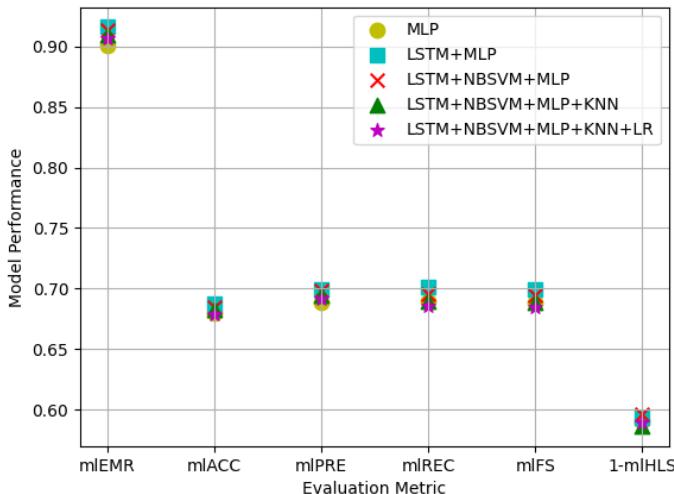


Figure 7.7: Best performing model of each training round for threat classification  
(Figure reproduced from Jiang and Atif (2021))

#### 7.6.4 CVSS SEVERITY CATEGORIZATION RESULTS

This experiment involves six separate classification tasks for all the CVSS v2 characteristics. Each classification task applies 5-fold stratified cross-validation for selected models. The evaluation process applies 6 metrics that are implemented with the `sklearn.metrics` packages, i.e., `balanced_accuracy_score` package for baACC; `confusion_matrix` and `sklearn.utils.multiclass` for maPRE, maREC, and maFS; `cohen_kappa_score` for maCKS; `hamming_loss` for maHLS. These metrics are introduced earlier in Section 7.4.3,

and are chosen considering the imbalanced classes of CVSS metrics.

Hard voting, or majority voting, is used for CVSS-characteristic classification and score prediction, in contrast to the soft voting technique used for threat classification. Majority voting works when the amount of base learners is equal to or bigger than three. In four rounds, we generate 21 performance files for each CVSS characteristic classification or score prediction. Figure 7.8 illustrates the main results, i.e., the best performing models of each training round. Figure 7.9 presents more details of the average performances of the trained models of each round. The narrative evaluation of the acquired results follows.

- In *AccessVector* classification (Figure 7.8(a) and Figure 7.9(a)), the individual model NBSVM has the best prediction performance overall. Hence, NBSVM classifier is a strong classifier for AccessVector classification, and the other four classifiers as weak classifiers. The ensemble of these four weak classifiers performs much better than their individual performances. The average performance increases when the amount of base learners increases from 1 (or Round 1) to 3 (or Round 2), and from 3 (or Round 2) to 5 (or Round 4), but drops when the amount is 4 (or Round 3).
- In *AccessComplexity* classification (Figure 7.8(b) and Figure 7.9(b)), the individual model NBSVM and the ensemble of LSTM, NBSVM and MLP deliver very close performance, both of which outperform the other models. For metrics baACC, maFS, maCKS, maHLS, the average performance of the models increases when the amount of base learners increases from 1 to 3, and from 3 to 5, but drops when the amount is 4. Metric maPRE has better average performance when the amount of base learners increases. Metric maREC has the best average performance when the amount of base learners is 3.
- In *Authentication* classification (Figure 7.8(c) and Figure 7.9(c)), it is clearly seen that the ensemble of base learners LSTM, NBSVM and MLP has the best prediction performance. Most of the metrics have better performances when the amount of participated base learners increases.
- In *ConfidentialityImpact* classification (Figure 7.8(d) and Figure 7.9(d)), the individual model NBSVM has the best prediction performance. But the ensemble of base learners LSTM, NBSVM and MLP also shows a strong performance.
- In *IntegrityImpact* classification (Figure 7.8(e) and Figure 7.9(e)), the ensemble of base learners LSTM, NBSVM and MLP has the best prediction performance. Individual models in Round 1 has the weakest average performance, while the ensemble model in Round 4 has the strongest average performance.
- In *AvailabilityImpact* classification (Figure 7.8(f) and Figure 7.9(f)), the ensemble of base learners, LSTM, NBSVM and MLP, has the best prediction performance. In general, the ensemble models in Round 2 have a sharp improvement in average performance compared to the individual models in Round 1.

The experimental results of threat and CVSS characteristic categorizations emphasize the need to utilize the multi-round ensemble paradigm, since it is unknown beforehand which round can deliver the best ensemble model, as illustrated in Figure 7.7 and Figure 7.8. Round 2 provides the best ensemble model for the authentication-categorization problem, for example. The best ensemble model for the confidentiality-impact categorization challenge is found in Round 1. In the threat categorization task, ensemble

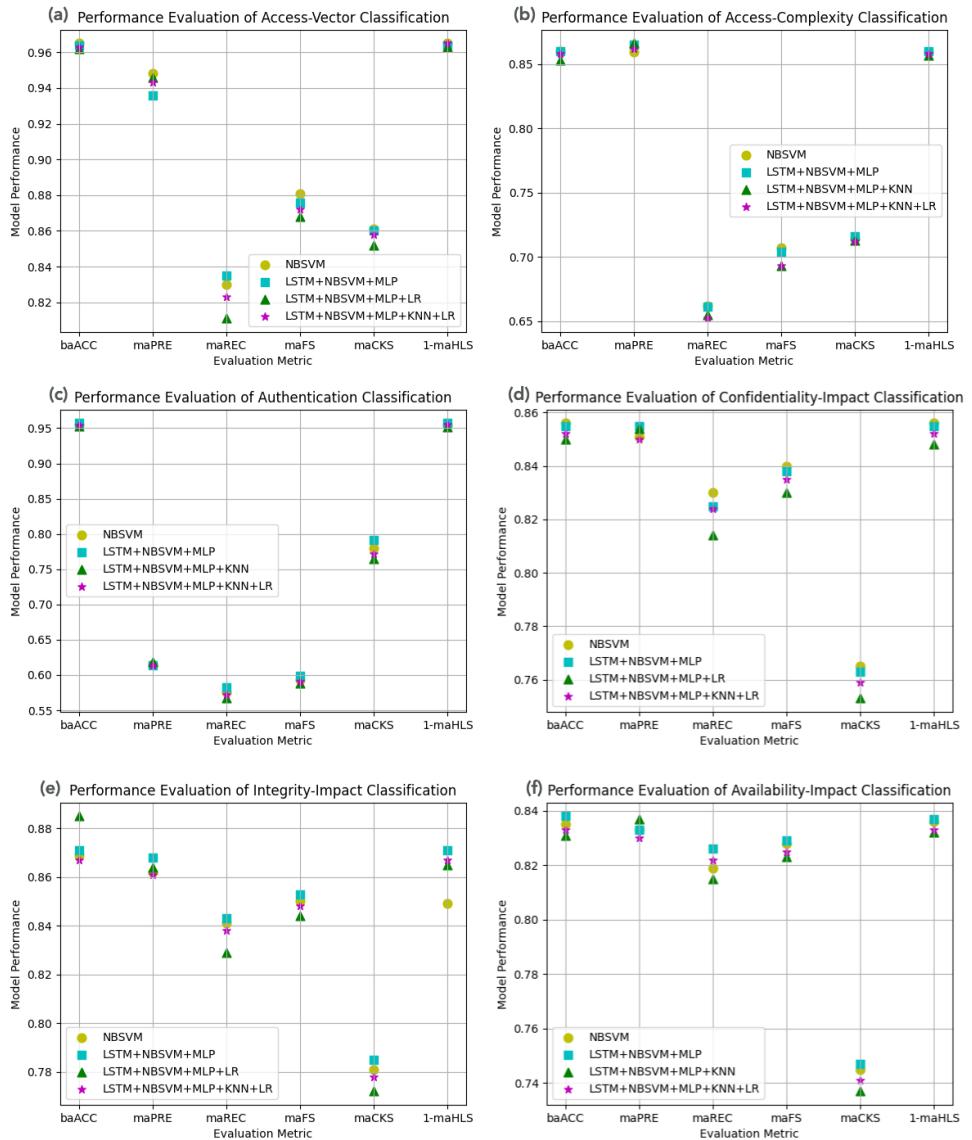


Figure 7.8: Best performing ML model of each training round for CVSS-characteristic classifications

(Figure reproduced from Jiang and Atif (2021))

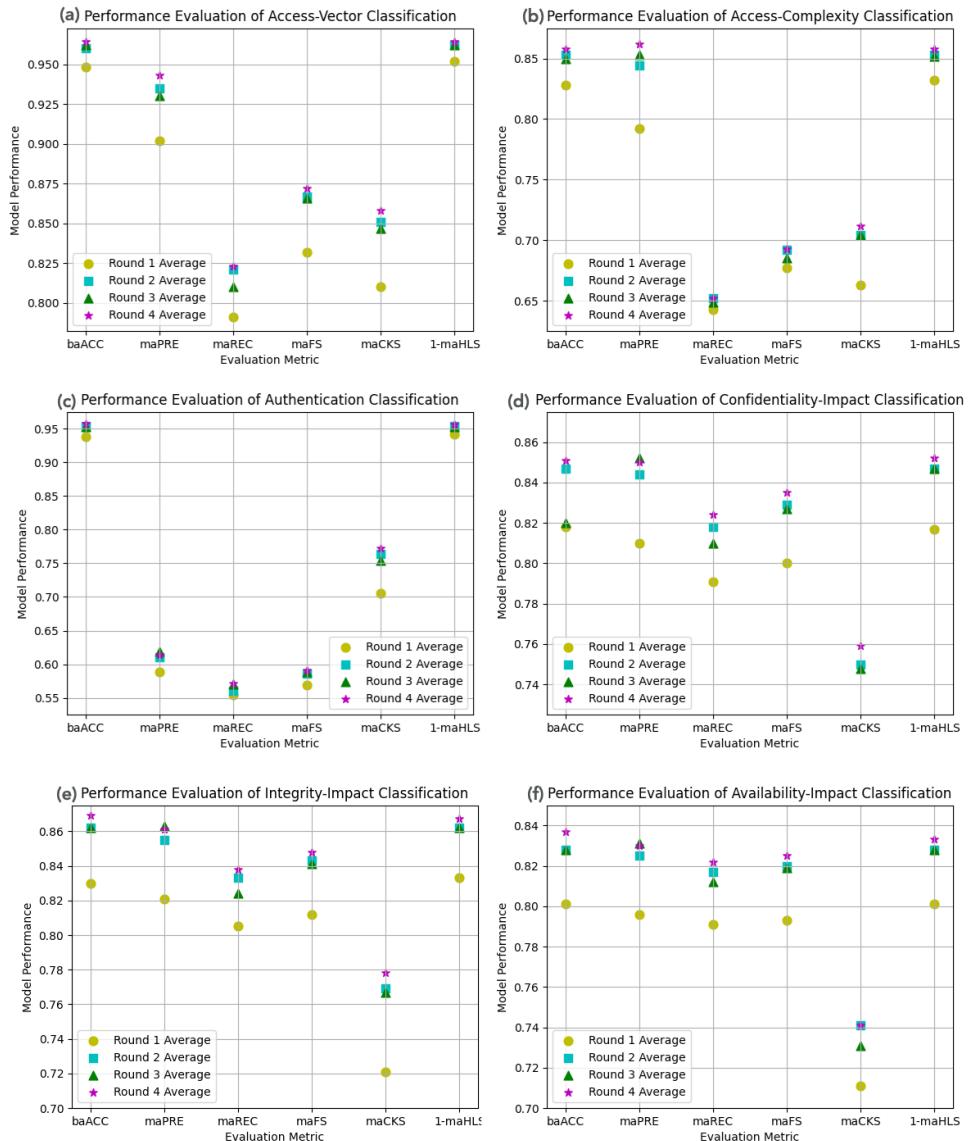


Figure 7.9: Average model performance in each training round for CVSS-characteristic classifications

(Figure Reproduced from Jiang and Atif (2021))

models from Rounds 2 and 3 fared better than models from earlier rounds. The experimental study demonstrates that our ensemble model selects relevant algorithms from five standard classifiers in order to improve their individual performance in the context of cybersecurity analysis.

The accuracy of *CVSS* v2 score prediction is computed considering the results of the above six classification tasks, using Equation 7.2 and using two values for  $\delta$  (0.5 and 0.05). And hence, the *CVSS* score accuracy is a harsh metric similar to Exact-Match-Ratio, which reflects the proportion of complete correct predictions of all the previous six classifications. Using the *CVSS* score accuracy metric, the ensemble of LSTM, NBSVM, and MLP has the best performance, as shown in Table 7.10.

Table 7.10: Evaluation of *CVSS* V2 score prediction on untrained dataset

(Table reproduced from Jiang and Atif (2021))

Model	<i>CVSS</i> -ACC( $\delta = 0.05$ )	<i>CVSS</i> -ACC( $\delta = 0.5$ )
NBSVM	64.58%	92.93%
<b>LSTM+NBSVM+MLP</b>	<b>64.75%</b>	<b>93.05%</b>
LSTM+NBSVM+MLP+LR	63.27%	92.81%
LSTM+NBSVM+MLP+KNN+LR	64.06%	92.90%

## 7.7 CONCLUSION

This thesis contributes to bridging the information the gap of missing data in vulnerability data sources, especially missing *CVSS* severity scores, *CWE* weakness classes, and *cvedetails* threat classes. Our proposed approaches also enhance the level of automation in the vulnerability assessment process and points out some limitations.

Identifying and assessing vulnerabilities are vital and challenging processes. We proposed increasing the efficacy of vulnerability-severity scoring systems that utilize *CVSS* standards to determine the severity of a reported occurrence of vulnerability. Our approach reconciles inconsistent vulnerability severity rankings contributed by multiple cybersecurity researchers and reduces potential conflicts originating from diverse *CVSS* techniques. We used majority voting to determine the score for inconsistent reporting of the same vulnerabilities in several vulnerability repositories. We then used these compatible vulnerability instances as ground truth to train a machine-learning model as a scoring basis. It is demonstrated in *Study VI* that the suggested model achieves excellent accuracy and micro F1-score thresholds in comparison to other studies. In addition, we applied our model in *Study VII* involving CI applications to demonstrate the automation of the suggested vulnerability score mechanism, which is used to minimize cybersecurity issues.

Furthermore, we suggest a ML-based weakness categorization method that predicts abstract weaknesses for vulnerability reports. The suggested weaknesses with high-level abstractions reduce the burden of navigating the whole *CWE* hierarchical tree structure for weakness allocation. Instead, security analysts can investigate the *CWE* entries that are members of the pinpointed classes. We initialized our approach with a bidirectional LSTM model together with *GloVe* embeddings. Our initialized model outperforms similar works towards *CWE* allocation automation in *Study VIII*.

Based on the vulnerability scorer and weakness categorizer, we further suggest a selective ensemble model that streamlines data integration, information processing and knowledge generation, to enable cybersecurity intelligence. This proposed ensemble mechanism uses a combination of different ML models, subsets of which are specialized in diverse vulnerability assessment tasks. We discuss the detailed process of how our proposed ML pipeline automatically finds out the best ensemble model for different vulnerability classification tasks. More precisely, our ensemble-based approach enables context-aware data analysis that aids situation awareness. In doing so, we empower security operators involved at various SOC levels with a localized and synchronized database that fetches data from several online vulnerability information sources. We resolve conflicting vulnerability-severity scores and diverse terminologies used by different parties, before adopting the discovered vulnerability instances as the training ground truth. We evaluate the proposed ensemble paradigm through an experimental analysis (*Study IX*) that involves five commonly used text-mining models for vulnerability description analysis and classification. This comparative study shows that our ensemble model has better performances than singular baseline ML models in the cybersecurity scenario of threat categorization and *CVSS* severity scoring. Our model is novel in terms of threat type categorization using security description and *cvedetails* threat labels. This exercise provides also means to adjust security investments at various organizational levels.

To summarize, this chapter presents methods that utilizes ML algorithms to automatically predict *CVSS* score and sub-metrics such as authentication, attack vector and confidentiality impact, as well as threat type labels and abstract *CWE* labels. In doing so, we discover missing labels in the vulnerability data sources. Next, we discuss how to combine our vulnerability database and CI system knowledge to serve vulnerability analysis of such intricate systems.

# MODELING CRITICAL INFRASTRUCTURE FOR VULNERABILITY ANALYSIS



# CHAPTER 8

## MODELING CRITICAL INFRASTRUCTURE FOR VULNERABILITY ANALYSIS

The need to improve the cybersecurity of CIs, through holistic system modeling and vulnerability analysis, cannot be overstated. This is challenging since a CI incorporates complex data from multiple interconnected physical and computational systems. Exploiting vulnerabilities in different systems leads to various cascading effects due to interconnections between systems. The previous chapters present data-driven methods that help find and analyze vulnerabilities in individual components. These methods provide a solid foundation for a vulnerability assessment of the whole system.

This chapter addresses the research question about how to model CIs to allow vulnerability assessment, considering the cyber-physical interconnections and dependencies within CIs.

This thesis delivers an extensible taxonomy and its usage in modeling such interconnections and the implied dependencies within complex CIs, bridging the knowledge gap between IT security and OT security. More specifically, the complexity of CI dependence analysis is harnessed by partitioning complicated dependencies into cyber and cyber-physical functional dependencies. These defined functional dependencies further support cascade modeling for vulnerability severity assessment and identification of critical components in a complex system. In addition to the proposed taxonomy, this thesis suggests power-grid reference models that make the proposed taxonomy easier to use and reproduce.

*Study X* validates the structural, functional adequacy, compatibility, operability, reliability, and maintainability characteristics of the proposed artifacts using the instantiated power-grid models. These evaluation metrics are presented in sub-section 4.3.5. Combining these instantiated modules, the proposed taxonomy can be used as a domain-specific language for dependence analysis and vulnerability exploit cascading modeling.

This chapter covers, but is not limited to results from *Paper I* and *Paper VI*. Partial results from *Paper VIII* are also covered.

### 8.1 ARTIFACT III-A: A TAXONOMY FOR CRITICAL INFRASTRUCTURE VULNERABILITY ANALYSIS

Figure 8.1 illustrates the interconnections between our proposed taxonomy and instantiated models. We categorize the various levels of our models based on the meta-modeling layers described by Jeusfeld, Jarke, and Mylopoulos (2009). A meta-model consists of formal assertions that clarify semantically connected model classes. Our vulnerability-driven CI taxonomy includes high-level classes such as *Component* and *Vulnerability*, along with shared class-level methods, properties, and constraints. Following generic constructions, constraints, and rules for each application domain, we developed models for the power grid and manufacturing application domains based on the taxonomy.

Then, we employ these models to generate instances that adhere to the same constructs.

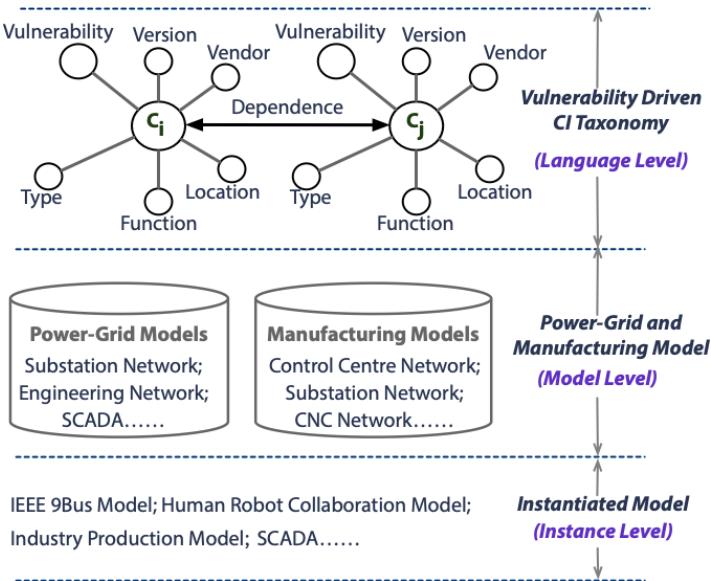


Figure 8.1: Connections between taxonomy and the instantiated models

Figure 8.2 provides an overview of our proposed taxonomy that fulfills critical infrastructure vulnerability assessment. This UML based overview builds on top of our cybersecurity meta model presented earlier in Figure 2.1 in Chapter 2, and links across three main types of objects, namely *CI*, *Security* and *Actor* objects, as discussed in the following text. Note that properties of modeled objects are partially shown in Figure 2.1 to allow readability. We focus on *CI* and *Security* objects in this thesis.

- *CI* object aggregates CI assets and related components. Systems and software components of various assets in the digitalized industry are interconnected. Vulnerabilities emerge due to these interconnections. A component can be either a software (e.g., a CAD program), a hardware (e.g., a milling machine), an Operating System (OS), or a network (e.g., a TCP/IP protocol).
- *Security* object accumulates threat, vulnerability, attack and remediation information and their relationships. Vulnerabilities may be exploited by an *attacker* (i.e., threat agent) in various ways using *exploits*. Each *vulnerability* instance has specific *impact*, *scope* and *exploitability* properties. Each *exploit* may be used to maliciously disrupt a targeted *component* in *CI* infrastructure in different ways.
- *Actor* object includes *CI staff* and *attacker*. Each instance of *Actor* object may further have a profile including identity information (e.g., role and responsibility) and character (i.e., grouping and sophistication). *Attacker* has specific *technique*, *procedure* and *tactic* used to trigger attack instances, by adopting enumerations like ATT&CK.

Next, we expand models of *CI*, *System* and *Actor* objects.

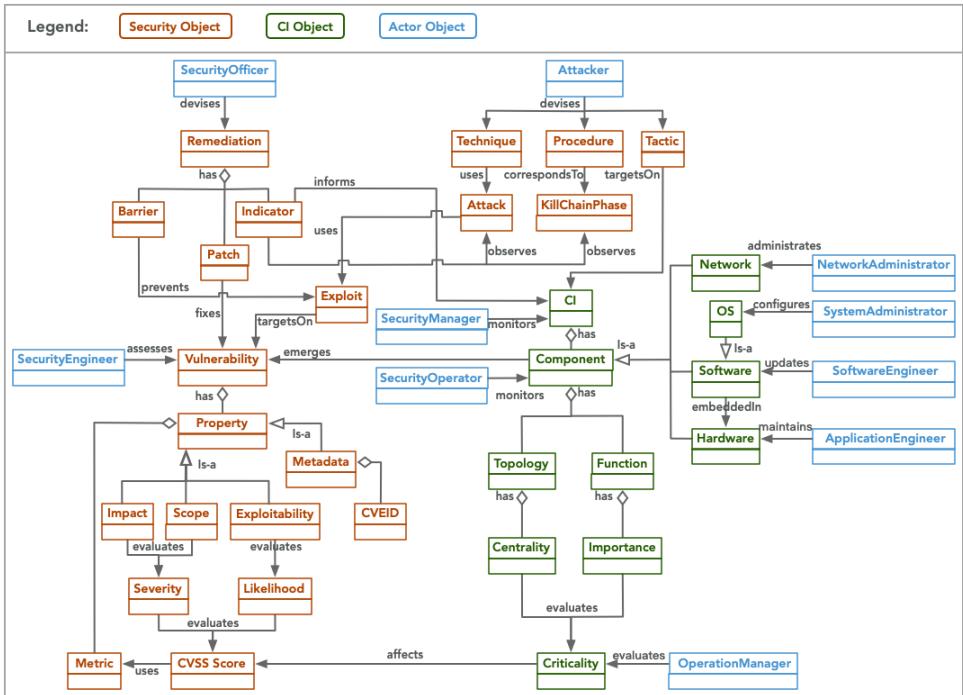


Figure 8.2: Overview of critical infrastructure vulnerability taxonomy  
(Figure reproduced from Jiang et al. (2019))

### 8.1.1 MODELING CYBER-PHYSICAL SYSTEM BASED CRITICAL INFRASTRUCTURE

Taxonomy is a representation of a domain that describes the objects within it. In the taxonomy of the CPS domain, component types constituting a cyber-physical system fabric can be identified as concepts. This taxonomy categorizes CPS elements into semantic modules that are used to construct a CPS model as a network of cyber and physical components.

Figure 8.3 illustrates the top-level structure of our taxonomy that starts with “*ComplexObject*”. *ComplexObject* is the most general class, and can subsume any object. For instance, we define a sub-class *Component*. *ComplexObject* has a relation *objProperty* to *Proposition*, which is used to attach various properties to power-grid components. Meanwhile, *ComplexObject* may further contain another *ComplexObject*. A component has two relations, namely data connections and sub-component configuration. Data connections refer to specific flows like data flows or control flows that bridge two given components. Besides, a component is decomposed into sub-components. For example, a RTU device is decomposed into the hardware and the embedded firmware.

Figure 8.4 depicts the taxonomy of components, including physical-, cyber-, and network-components. Cyber components are integrated into physical components that are located in specific geographic areas. Consequently, both physical and embedded cyber components share the same physical connections. A network component is an organization where a specific collection of components adhere to a shared set of access and man-

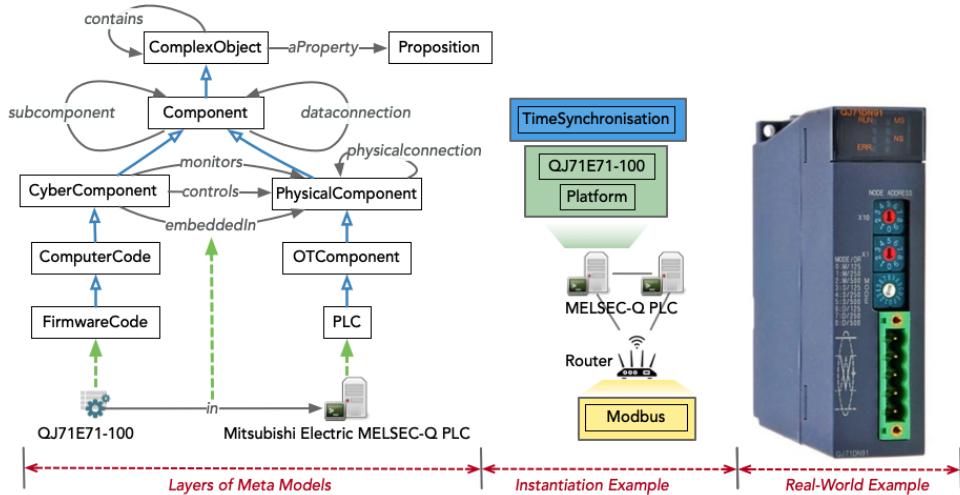


Figure 8.3: Top level cyber-physical taxonomy and instantiation example  
 (Subclass relations are denoted by blue arrows with white arrow heads.  
 Instantiating relations are visualized by green broken links. Figure reproduced  
 from Jiang, Jeusfeld, and Ding (2022))

agement rules. Our taxonomy includes facts, constraints, types, and attributes pertaining to security. For example, we define each component's properties to clarify its vendor, product model, version, build number, and protocol. We may instantiate a *MELSEC-Q PLC* (2022), provided by the vendor *Mitsubishi Electric*, with adopted protocol *Modbus*.

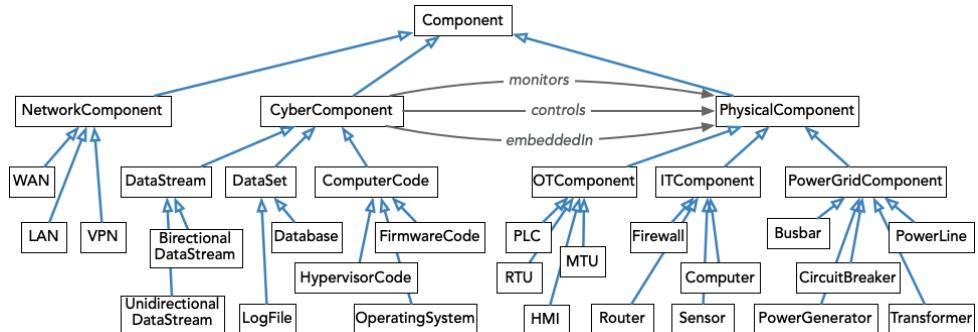


Figure 8.4: Cyber, physical and network components in the taxonomy  
 (Figure reproduced from Jiang, Jeusfeld, and Ding (2022))

Our taxonomy further extends the physical component to include a variety of IT and OT components employed by SCADA and security information and event management (SIEM) services at the cyber and control layers. The embedded cyber components distinguish IT, OT, and power-grid components. The SCADA control system is supported by RTU, MTU, HMI, and sensors, as well as routers and optical networks (Humayed et al., 2017), as introduced in sub-section 2.1.2. We categorize these components as OT components. Firewalls and endpoint security tools support SIEM's data analysis and cor-

relation, both of which belong to IT components (Vielberth et al., 2020). In particular, MTU periodically begins and acquires RTU data and enables remote control activities to be performed by operators. RTU takes field data such as process data and variables directly from sensors and deploys commands via actuators. HMIs may be freestanding terminals or incorporated within other devices, such as MTUs. RTU and MTU are linked to other SCADA components, such as SCADA servers, via routers, fiber optic cables, and switches (Boyer, 2009)(Stouffer, Falco, Scarfone, et al., 2011).

Cyber components subsume computer code and data sets captured at the cyber-layer level. SCADA programs are integrated in microcontrollers to monitor physical power-grid processes, for instance (Boyer, 2009). Components of computer code represent the actual code being executed and embedded in physical components. Additional examples of computer code components include firmware code, operating system code, hypervisor code, and so on. Firmware code typically runs on the chip's bare metal and supports low-level hardware control. One example is HMI firmware that has graphical libraries in which graphical symbols with tag names are associated with particular devices and parameters of the devices, such as a specific switch and its ON/OFF status. An operating system manages the central hardware of the host computer and supports hardware-software interactions. The code of the hypervisor virtualizes the hardware that executes kernel-model processes. The setup between an operating system and a hypervisor can also be specified as bare-metal or hosted hypervisors.

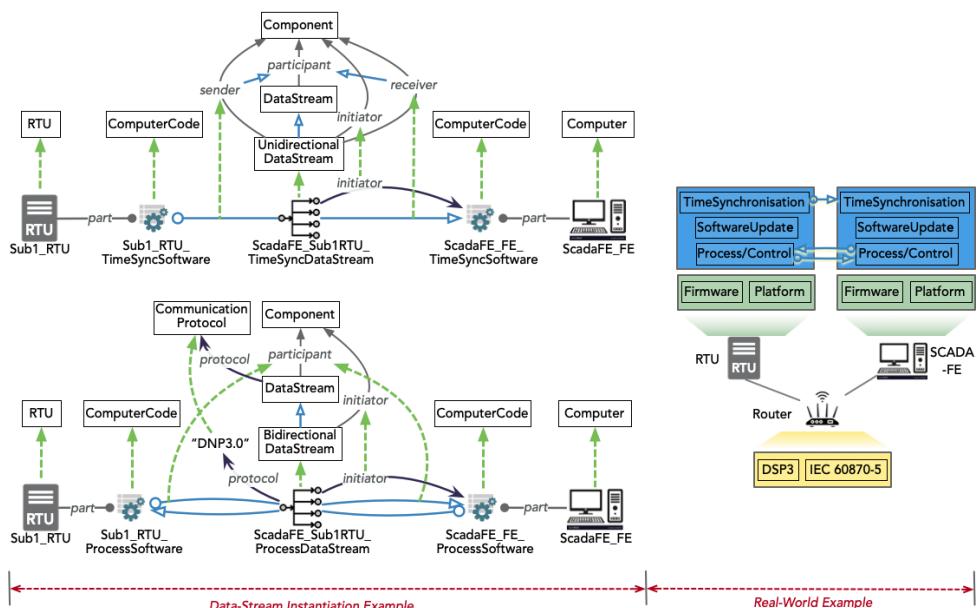


Figure 8.5: Unidirectional and bidirectional data stream example  
(Figure reproduced from Jiang, Jeusfeld, and Ding (2022))

Cyber components further subsume the data stream illustrated in Figure 8.5. In addition, the data stream subsumes both bidirectional and unidirectional data streams. Data stream is a critical concept in our taxonomy through its contribution to the system dependencies. A data stream object must include at least two components as participants, with one of those components serving as the initiator. Typically, two parties commu-

nicate using a master-slave method, with a master device initiating queries and a slave device responding with requested data to complete transactions. One participant acts as the sender and the other as the receiver for a unidirectional data stream. In contrast, the sender of a bidirectional data stream operates as the receiver in the opposite direction. The data stream specifications are particularly relevant for the study of SCADA automation and cybersecurity (PES, 2008).

A corporate or communication network consists of relay stations such as routers, switches, and firewalls, as well as endpoints such as computing servers. Routers and switches typically have access to the majority of network segments and occupy prime data exfiltration positions. Switches are responsible for processing and managing numerous Layer 2 protocols, which are often enabled by default on all accessible ports. A network component adheres to a particular protocol, which is a set of rules, syntax, and semantics that enables the exchange of data between two or more entities. Wide area network (WAN), local area network (LAN), and virtual private network (VPN) are also network components.

### 8.1.2 MODELING VULNERABILITY AND OTHER SECURITY RELATED OBJECTS

On top of the security data source objects and their attributes briefly mentioned earlier in Figure 6.2 and Figure 6.7 in Chapter 6, we further define *Vulnerability* that is subsumed under *ComplexObject*.

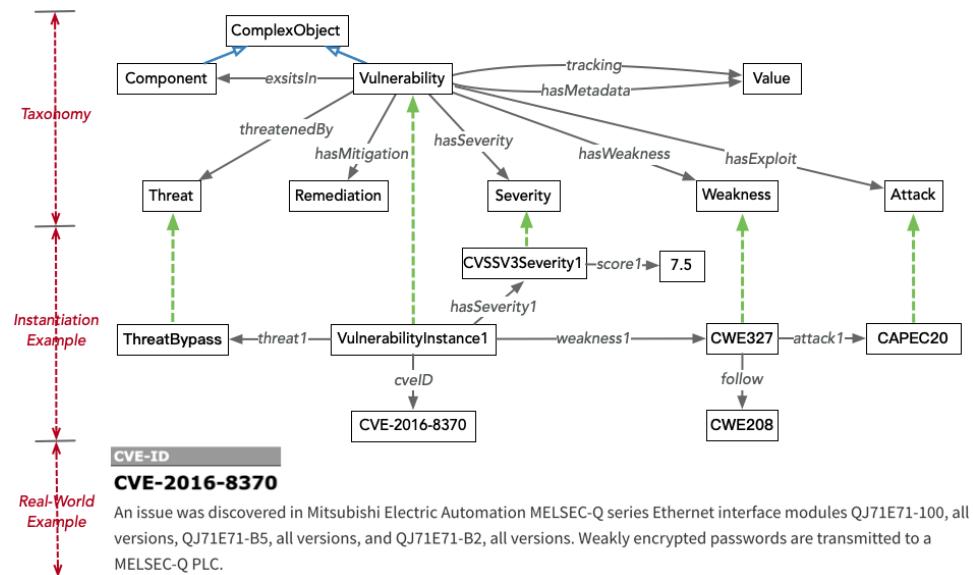


Figure 8.6: Taxonomy of Security Objects

*Vulnerability* exists in *Component* that matches the product configuration affected by this *Vulnerability*. *Vulnerability* further has attributes, including metadata, tracking, weakness information, severity, threat, related attack and corresponding remediation. Figure 8.6 illustrates these related attribute of *Vulnerability* as well as an instantiated example with *CVE-ID* as *CVE-2016-8370*. This vulnerability instance is assigned a weak-

ness *CWE-327* that further follows or is resultant from *CWE-208* and may be exploited by attack *CAPEC-20*.

### 8.1.3 MODELING STAKEHOLDER ACTORS

As introduced earlier, *ActorObject* includes *CI-Staff*, and *Attacker*, as illustrated in Figure 8.7. *CI-Staff* instantiates *SecurityOfficer*, *SecurityManager*, *SecurityOperator*, *SoftwareAdministrator*, *ApplicationEngineer*, *NetworkAdministrator*, etc.,. *SecurityOperator* monitors corresponding *Assets*. *SecurityManager* monitors *CI Indicators*. Conceptually, *SecurityOfficer* needs to gauge budget investments through adopting an expert-system interpretation of numerical vulnerability-indicators to carry out mitigation decisions like vulnerability-patching. *SecurityEngineer* rank assets by using following vulnerability-indicators : (i) their *Criticality* evaluated by *OperationManager*; (ii) their vulnerability exploitability *Likelihood* identified by *SystemAdministrator*, *SoftwareAdministrator*, and *NetworkAdministrator*, as well as (iii) the impact *Severity* of threats defined by *ApplicationEngineer* (Kure, Islam, and Razzaque, 2018). By distributing dynamic vulnerability-management tasks throughout CI organisation, we argue that it improves the level of communication between vulnerability-handling stakeholders.

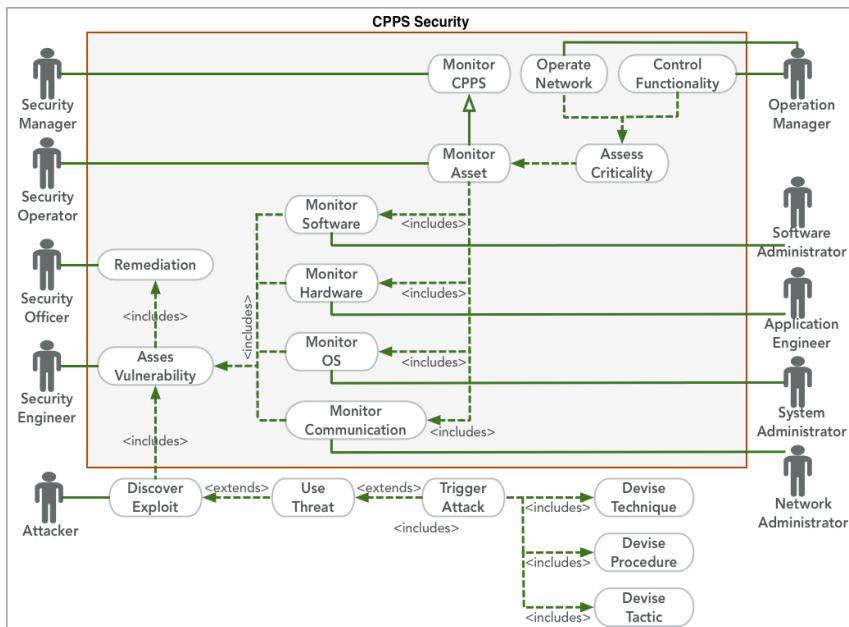


Figure 8.7: CI security user interaction model  
(Figure reproduced from Jiang et al. (2019))

For example, *SecurityOperator*, *SecurityManager* and *OperationManager* are involved to measure temporal or environmental metrics. Vulnerability instance *CVE-2015-0997* has a CVSS V2 base score of 3.3. According to the v2 documentation introduced earlier in Section 2.2.1, relevant actors are involved in the analysis process, as shown in Figure 8.8. Taken into consideration of the given temporal and environmental measurements, a final CVSS v2 score of 3.2 is assigned. Application specialists can also provide valuable information such as potential costs for system recovering once success attacks happen.

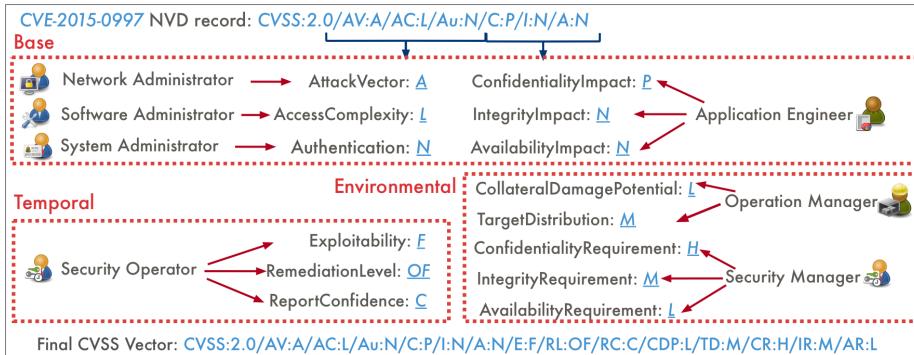


Figure 8.8: Quantitative vulnerability assessment involving stakeholders  
(Figure reproduced from Jiang et al. (2019))

## 8.2 ARTIFACT III-B: CYBER AND CYBER-PHYSICAL FUNCTIONAL DEPENDENCE

We further propose a model-based security engineering method that has our taxonomy in the central role, to support dependence analysis and vulnerability assessment.

### 8.2.1 DEPENDENCE RULE

Here we define functional dependencies (or FD) as: If component  $C_i$  depends on component  $C_j$  to complete its functional activities properly, then we say that component  $C_i$  has functional dependence  $FD_{(i,j)}$  on component  $C_j$ . We further define seven FD rules as depicted below, which are employed to describe the complexity of a software component. These seven rules can be used to define system dependencies using the static system configuration information.

1. **FD Vertical Rule  $V^1$ :** If a cyber component  $C_i$  is embedded in an IT or OT component  $C_j$ , then  $C_i$  is functionally dependent on  $C_j$ , or  $FDV_{(i,j)}^1$ .
2. **FD Vertical Rule  $V^2$ :** If hypervisor or operating system component  $C_i$  contains cyber component  $C_j$ , then  $C_j$  is functionally dependent on  $C_i$ , or  $FDV_{(j,i)}^2$ .
3. **FD Horizontal Rule  $H^1$ :** If an OT component  $C_i$  contains a cyber component  $C_k$  that collects process data from a physical component  $C_j$ , then  $C_k$  is functionally dependent on  $C_j$ , or  $FDH_{(k,j)}^1$ .
4. **FD Horizontal Rule  $H^2$ :** There exists control data from a cyber component  $C_i$  (embedded in an OT component  $C_k$ ) to a physical component  $C_j$ , then  $C_j$  is functionally dependent on  $C_i$ , or  $FDH_{(j,i)}^2$ .
5. **FD Data Rule  $D^1$ :** There exists data stream between two cyber components  $C_i$  and  $C_j$ , and  $C_i$  is the receiver of the data stream, then  $C_i$  is functionally dependent on  $C_j$ , or  $FDD_{(i,j)}^1$ .
6. **FD Data Rule  $D^2$ :** There exists data stream that listens to dataset  $C_j$ , and  $C_i$  is the receiver of the data stream, then  $C_i$  is functionally dependent on  $C_j$ , or  $FDD_{(i,j)}^2$ .

7. FD Network Rule  $N^1$ : If a server computer  $C_i$  is connected to a network through a router component (or a switch component)  $C_j$ , then  $C_i$  is functionally dependent on  $C_j$ , or  $FDN_{(i,j)}^1$ .

All rules are implemented as deductive rules in the ConceptBase system. Following these dependence rules, we conduct some static analysis on *SCADA* and *Substation* based on the aforementioned reference models. The red dashed lines highlight the functional dependence. Partial SCADA network contains *SCADA\_Historian* and *SCADA\_Server*, *SCADA\_FE* workstation and a *RTU* in one substation, as illustrated in Figure 8.9.

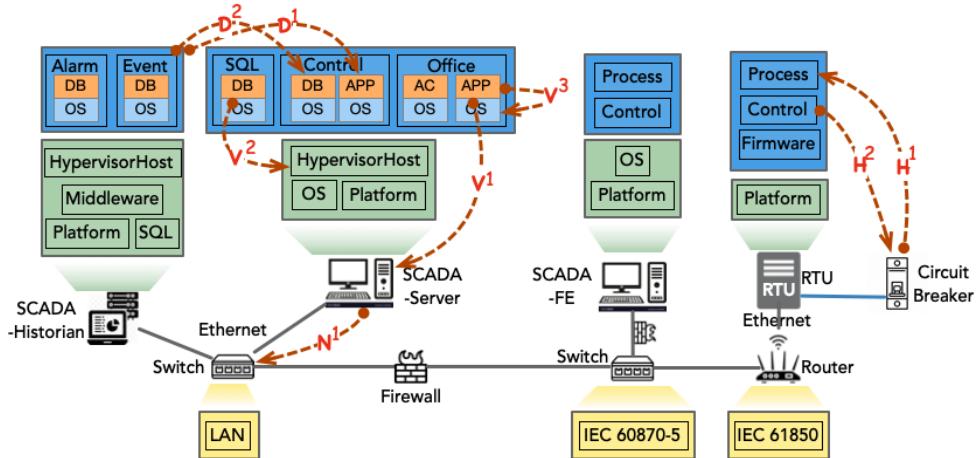


Figure 8.9: Dependence analysis example of a substation RTU

Hardware components connect with operating system applications and manage PC storage on the physical server. In the meantime, a hypervisor host deploys and serves virtual systems, providing a virtualization abstraction layer. The hypervisor in *SCADA Server*, for instance, provides virtual machines for three system packages: *SQL*, *Control* and *Office* systems. The *SQL* system contains a database engine that processes queries and manages database files. The *Control* system stores and retrieves power-grid process and control data using queries. The *Office* system provides maintenance and service to the power-grid software, firmware, and configurations. According to the vertical FD rules, the application (APP) server in the *Office* system is vertically dependent on the guest operating system (marked as  $V^3$ ) and the physical server (marked as  $V^1$ ). The database (DB) server in the *SQL* system is vertically dependent on the hypervisor host (marked as  $V^2$ ).  $H1$  and  $H2$  represent the dependencies between *RTU* and *CircuitBreaker* that are caused by process and control data.

Figure 8.9 highlights the historical analysis data stream example marked as  $D^1$  and  $D^2$ .  $D^1$  illustrates the dependence of the data stream receiver, the *Event* system in *SCADA\_Historian*, towards the data stream sender, the application server in *Control* system of *SCADA\_Server*.  $D^2$  presents the dependence of the same receiver on the listened database.

We discuss another example to illustrate functional dependencies in manufacturing system, as presented in Figure 8.10. Programming instructions (e.g., G-code or M-code files) are transmitted from the computer aided manufacturing (CAM) server to the control server in the control center, which is marked as  $D^{1-1}$ . The machining data is further

sent to the PLC controller for production purpose, reflecting dependence  $D^{1-2}$ . Meanwhile, process data and duplicated machining data are sent to and stored in the historian server, labeled as dependence  $D^2$ . The historian server manages a database of time-tagged data points about the production system. PLC controller is directly connected with the PLC gripper, the monitoring and controlling processes of which reflect dependencies  $H^1$  and  $H^2$ . Meanwhile, CAM servers are connected to the control server through switches and routers, which shows dependence  $N^{1-1}$ .

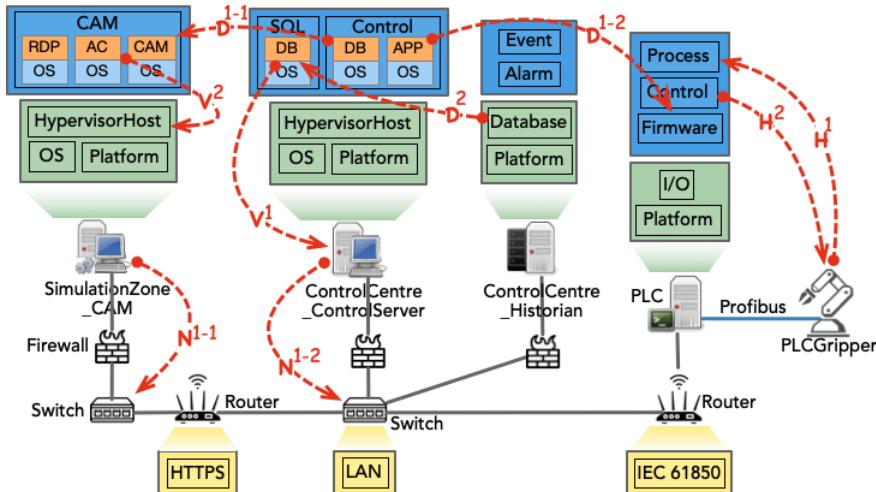


Figure 8.10: Dependence analysis example of manufacturing system

On top of defined correlations between component nodes, we further define dependency matrix  $FD_{(i,j)}$  between  $C_i$  and  $C_j$ . Such a dependency matrix supports analyzing nodes' centrality and influence levels. Meanwhile, dependence rules assist CPS cascade modeling, which is introduced next.

### 8.2.2 CASCADING MODELING AND CRITICALITY ANALYSIS

Cascading is the propagation behavior exhibited by a series of system events or failures. Failure is the state or condition of not achieving a desired or intended goal, and it can be caused by external forces such as an attack or the failure of neighboring components. Failure might occur on any component or on many components. A cascading failure begins someplace in the system, causing a subsequent failure in another component (Guo et al., 2017) (Vaiman et al., 2012). We further define the following rule to support cascade modeling, which are extended to the transitive closure: “There exists a failure or compromise of a component  $C_i$  that a component  $C_j$  is functionally dependent on, then the failure would probably propagate to  $C_j$ ”.

Here, we assert that the transmission of a failure from  $C_i$  to  $C_j$  has a specific probability, given that system configurations or network structures with adequate security compliance lower this probability. In *Study VI* and *Study VII*, we assume that this probability is equal inside the system and leave weighted probability analysis for future research..

We calculate the number of components that have direct functional dependence on component  $C_i$ , and define it as  $N_{i,j}^{FD}$  where  $0 < i, j < M$  ( $M$  is the number of components).

Such a component is a critical function point with higher criticality.

## 8.3 ARTIFACT III-C: REFERENCE MODELS OF POWER GRIDS

Our taxonomy can be expanded to increase expressiveness within a particular domain. We follow the Purdue (Williams, 1994) enterprise reference architecture model when instantiating objects in the cyber and control layers. The Purdue Model is adopted in internal standards such as *IEC/TS 62443-1-1:2009(E)*. Even though the Purdue Model is used more in manufacturing architectures (Boyes et al., 2018), its structure still applies to the similar cyber and cyber-physical layers in smart grid. We also incorporated our knowledge about the smart-grid architecture, especially the physical layer and its related components and processes, that are gained through our organized workshops, newspaper articles, and discussions with a municipal electricity grid company in Sweden.

The figures and code samples included in this section are based on ConceptBase (Jarke et al., 1995), respectively the O-Telos dialect of the Telos knowledge representation language (Mylopoulos et al., 1990) (Koubarakis et al., 2021).

Figure 8.4 illustrates some partial examples of cyber and physical components in our power-grid taxonomy. We specify that physical components further encompass power-grid components that are employed for electric power generation, transmission, transformation, and distribution by taking into consideration the functionality of power-grid systems. A physical component has a geographical location or a particular position. Meanwhile, components of the power grid have power connections. A circuit breaker is an example of a power-grid component that is used to disconnect a power transmission. Another example is a transformer that transfers electric power between two electric circuits. In addition, we specify functional criteria for power-grid components such as voltage and connected power lines.

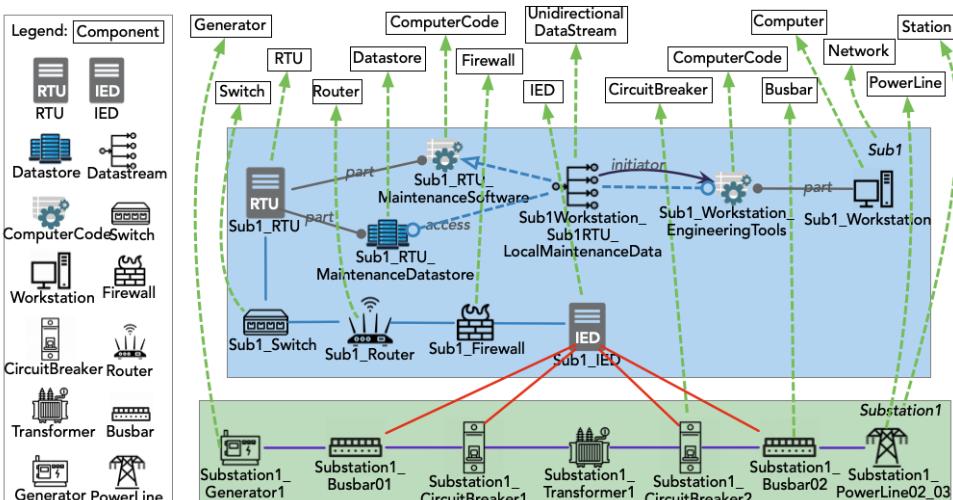


Figure 8.11: Instantiated power-grid substation

Figure 8.11 shows a power substation as an instance of our power-grid taxonomy. The

instances are partially shown to ensure readability. Green dashed arrows represent instantiating. Grey solid lines and purple solid lines represent data connections and power connections, separately. Light grey lines marked with "part" represent system configurations, meaning one component is subsuming the other component. A power generator *Generator1* has power connection with *Busbar01* which is connected to the second busbar *Busbar02* through a transformer *Transformer1*. *Busbar02* also has power connection with power line *PowerLine02\_03*. On top of *Substation1*, an operation network *Sub1* covers the power process control and monitoring. IT components (i.e., *Sub1\_RTU* and *Sub1\_Workstation*) are connected to the power-grid components (i.e., *Busbar01* and *Busbar02*) through data connections (i.e., fibre). *Sub1\_RTU* and *Sub1\_Workstation* are also connected through a data stream for local maintenance. This unidirectional data stream has access to the maintenance data store embedded in *Sub1\_RTU*.

Cyber, control, and physical network levels are modeled to uncover inter-dependencies between CIs. Each layer of CI networks comprises distinct functional portions or zones. Network zones are connected through routers and are protected by firewalls, as discussed in the next sub-section.

### 8.3.1 PUBLIC INTERNET AND OTHER NETWORKS

The cyber layer includes a general internet area and the power-grid enterprise network. The wide internet area contains a *CustomerService* network, an *Analyser* network, a *Vendor* network and an *EnergySupplier* network, as illustrated in Figure 8.12.

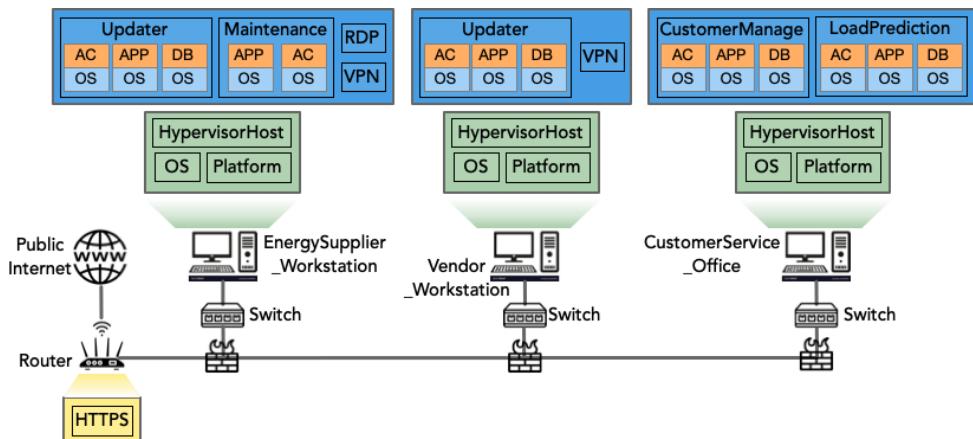


Figure 8.12: Public Internet and other networks

The *CustomerService* network contains a server computer embedded with two software packages, namely *CustomerManage* software that regulates customers' power consumption, and *Analyser* software that works for load prediction (Abubakar et al., 2017). These two packages may alternatively be combined into a single module, such as *ABB Energy Manager* (2022). Multiple vendors provide separate software and hardware to meet the diverse access, operational, and technological requirements of the smart grid. Some of these companies require privileged remote network access or VPN tunnels to support, manage, or debug particular smart grid technologies and systems (Zeinali and Thompson, 2021).

*EnergySupplier* network refers to external power suppliers' business administration and marketing management. Sometimes energy suppliers employ VPN or remote desktop protocol (RDP) access to remotely log into distributed energy resource (DER) substations for monitoring and operating purposes (Ying, Yirong, and Ning, 2014). *Updater* refers to the software package, input and output (I/O), cumulative update file, and other programs required to manage the updating of hardware, software, and firmware components. The *Maintenance* package includes the programs required to manage system configurations.

Typically, the *Vendor*, *CustomerService*, and *EnergySupplier* networks belong to separate stakeholders. These stakeholders gain access to the power-grid enterprise network via an intermediary, namely the *PublicInternet* network, which facilitates internet applications such as web browsing.

### 8.3.2 OFFICE, ENGINEERING, AND SECURITY OPERATING CENTER NETWORK

The enterprise network contains an *ITAdministration* network for IT administration management, an *Engineering* network for system maintenance and configuration update, a *SOC* (security operation center) network for security-related analysis and safety inspection, as well as an *Office* network for local office operation (see Figure 8.13).

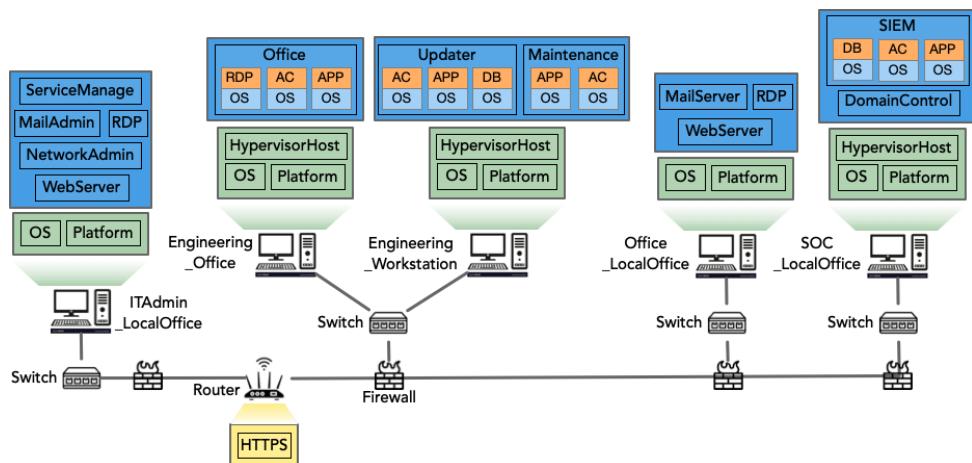


Figure 8.13: Office, engineering and security operating center networks

The *ITAdministration* network is connected to public internet servers via firewalls. *ITAdministration* network is further connected to *Engineering* network, *SOC* network and *Office* network through router and firewalls. The *ITAdmin*, or IT administration network, is responsible for network operation as well as preventing and resolving network issues locally or via RDP. In addition, network administration and mail administration are used individually to maintain and setup network and mail routing. The *SOC* network involves system monitoring and risk management, as well as control and digital forensics. A typical tool used in *SOC* is *SIEM*, which leverages advanced analytics for incident response and *SOC* automation (Vielberth et al., 2020). A local office server manages mail setup, remote desktop software, and the web browser on the *Office* network. *Engineering* network contains a local server that undertakes business-driven investigation

and SCADA statistic analysis. *Engineering* network also includes a workstation that oversees system software maintenance and updates.

### 8.3.3 CONTROL CENTER NETWORK

The control layer includes two networks, namely a control center and a *SCADA WAN*. *Control Centre* mainly involves *SCADA* for process data monitoring, control command distribution, and power process synchronization, as illustrated in Figure 8.14. The control center is connected to a variety of intelligent grid devices originating from power generation substations, high and low voltage transformation substations, distribution assets, and distributed controlling workstations. The *SCADA\_Server* monitors and controls these distributed substations (Knapp and Samani, 2013). The real-time power process data is virtually presented on *SCADA\_HMI* and then further transmitted from *SCADA\_Server* to *SCADA\_Historian* for statistical analysis. Furthermore, system update and maintenance data is transferred to and stored in *SCADA\_FTP* before direct usage in the controlling servers. *SCADA\_Timer* is in charge of the time synchronization of the whole system (Boyer, 2009) (Stouffer, Falco, Scarfone, et al., 2011).

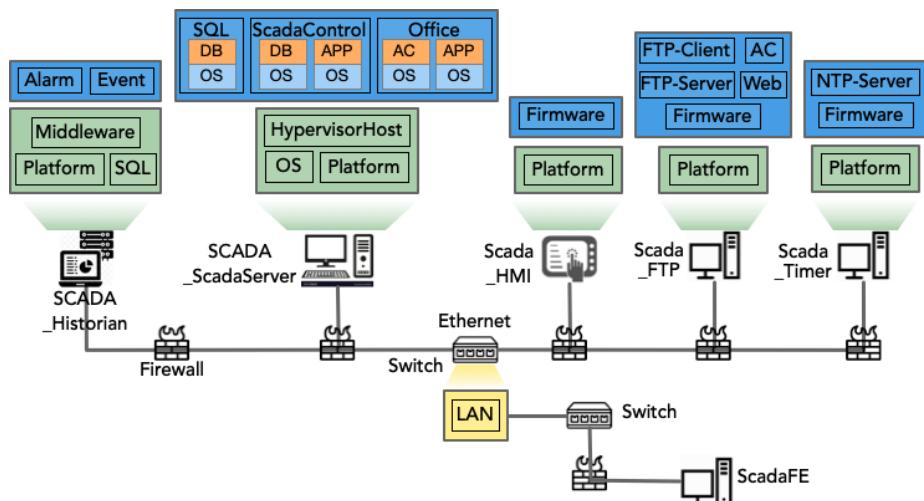


Figure 8.14: Control center network

*SCADA WAN* is a network shared by the *SCADA-FrontEnd (FE)* server and dispersed substation networks. The *SCADA-FE* server manages event-based communication with field devices and is therefore responsible for data processing and transfer management. Specifically, *SCADA-FE* functions similarly to a master station and frequently requests data from field devices such as RTUs.

### 8.3.4 SUBSTATION NETWORK

Figure 8.15 shows substation network includes LANs between RTUs and local workstations, as well as *Bay network* that lies in the interface of the control layer and physical layer. *Bay control IEDs* provide flexible control and backup protection for physical components such as circuit-breakers and earthing switches (Brand, Wimmer, and Lohmann,

2003), which normally follows the *IEC 61850* communication protocol (Brand, Brunner, and Wimmer, 2011). In the physical layer, *IEDs* (intelligent electronic devices) and *SIS-PLCs* (safety instrumented system PLC that can enable emergency shutdown) are connected to RTUs. The data connections construct the link between control units and physical units, enabling RTUs at local substations to remotely control and monitor power processes. These control and monitoring features include devices switch, set-points for generators, and sequential control.

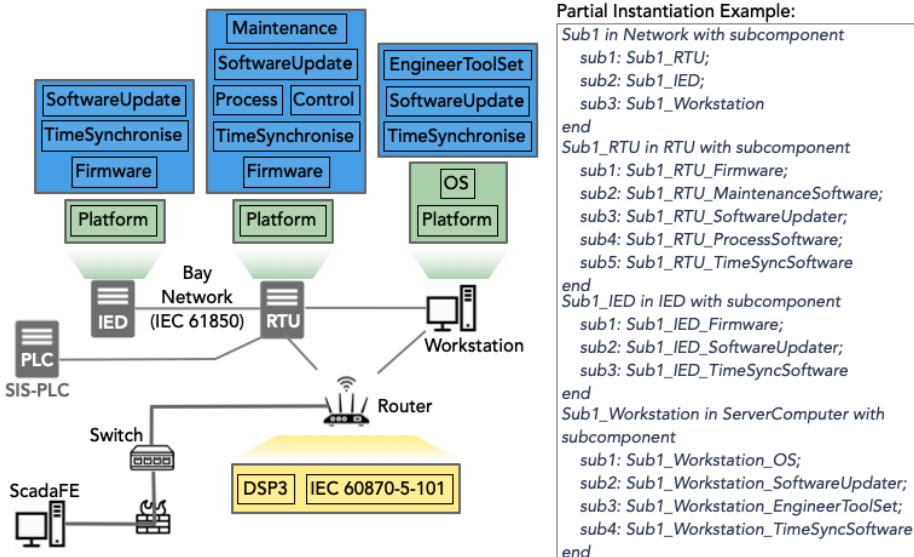


Figure 8.15: Substation network

### 8.3.5 POWER-GRID SUBSTATION

The transmission substation is connected to a power generator. This substation contains six circuit breakers, two transformers that convert between two transmission voltages, as well as several transmission lines, busbars, and switches. To be more explicit, two high-voltage switches enable the neural line designated as *NLine1* to be disconnected from the rest of the network and linked to a grounding system such as a ground fault neutralizer (see Figure 8.16 (a)).

Figure 8.16 (b) presents similar structures between the transmission and distribution substations (Ruland et al., 2017), namely two or more transmission lines as power input, feeders as power output, and one or two transformers in the middle. Meanwhile, smart meters are deployed to record electric energy consumption, voltage levels, and other physical process data (Korman et al., 2016). In addition, a communication network supplies this electrical grid with supervisory process management. Such a communication network consists of many dispersed LANs connected to the power substations and control center networks individually.

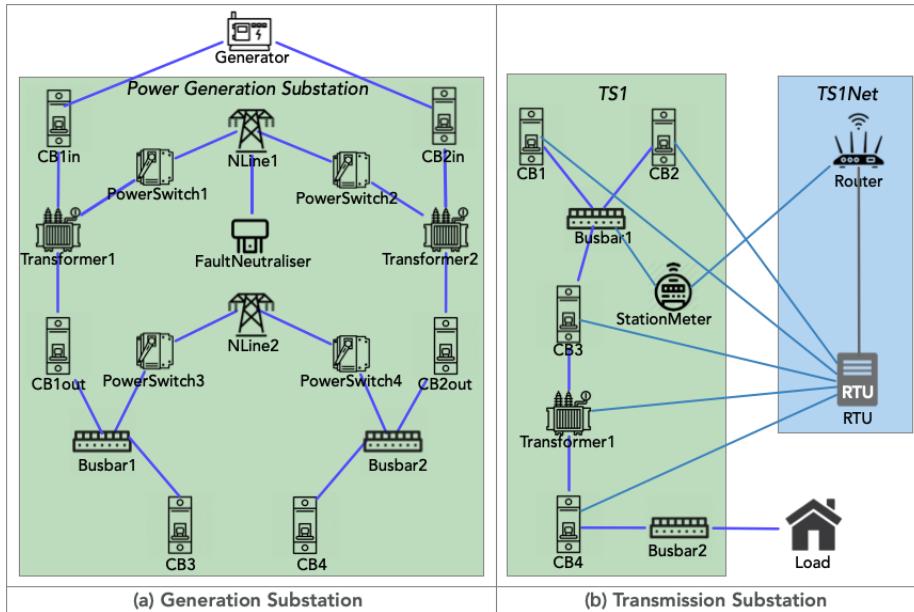


Figure 8.16: Power generation and distribution substations

### 8.3.6 DATA ASSET IDENTIFICATION

Identification of information assets is a vital step in the risk management process highlighted in ISO 27000 series (Disterer, 2013). The power system produces data that can be turned into valuable information when appropriately processed and encrypted. This information is a valuable asset that benefits optimized investments, accurate problem analysis, and safe utilization of the power system. This thesis focuses on the critical data assets of a power-grid system and the containers where the assets are stored, transported, and processed.

#### (i) Process data and process control data

Process data are measurements of power processes gathered by dispersed sensors. Figure 8.17 displays the process data frequently queried by the process liaison, SCADA FE, to the centralized system platform SCADA. In order to generate commands for the SCADA SystemServer, SCADA application servers, including the Analyzer server, simultaneously compute process data from real-time and historical databases. Then, these commands are communicated to distributed actuators to ensure optimal power flow. In the meantime, process data are transmitted to the safety-inspection server for inspectional analysis, such as evaluation of voltage stability. After acquiring and validating an unstable power status, the safety-inspection server sends SCADA alarms to the SIS-PLC for emergency power-grid shutdown.

#### (ii) Historical analysis and load-prediction data

Figure 8.18 illustrates SCADA process data that is inserted into the historical database with timestamps. Several servers, including an HMI server, file transfer protocol (FTP) server, historian server, and SCADA system server, are required for historical data anal-

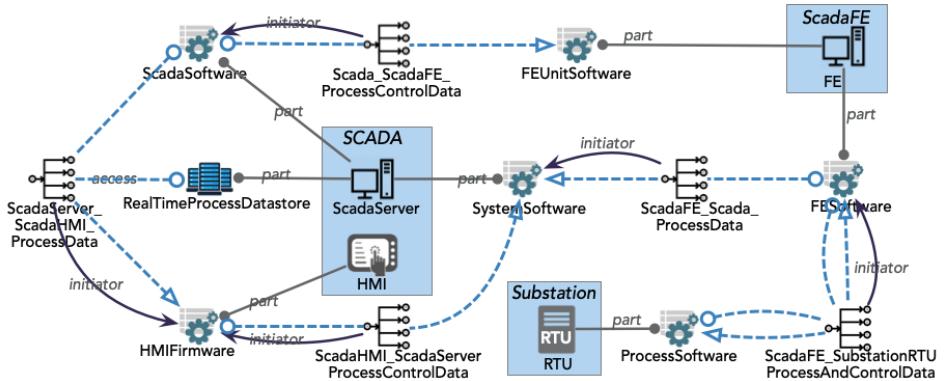


Figure 8.17: Process data and control commands stream

ysis in order to extract power generation and transmission trends. SCADA operators get historical data from historians and display the information on SCADA HMIs.

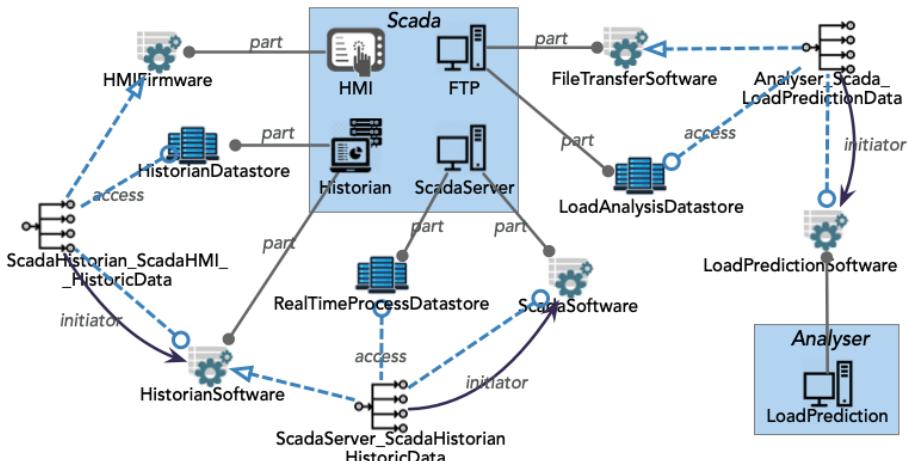


Figure 8.18: Historical analysis and load-prediction data streams

### (iii) Time synchronization data

Figure 8.19 presents time-synchronization data organized into synchronized instances sampled from several sources and involves a range of current sensors (Fang et al., 2011). RTUs and other OT components transport telemetry data from sensing devices to SCADA and generate time-synchronization data flows. The control loop is then completed by communicating commands from the master supervisory system's Timer server to the linked physical power components. Meanwhile, phasor measurement units (PMUs) measurements provide another common time data source for synchronization. These PMUs measure the magnitude of power signal wave (called phasors) across the smart grid, which is generally retrieved from a global positioning system (GPS) receiver (Cho, Shin, and Hyun, 2001).

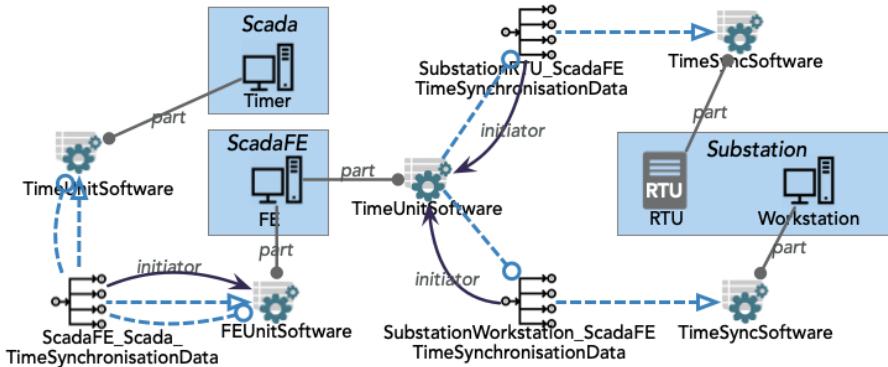


Figure 8.19: Time-synchronization data stream

#### (iv) System update data and configuration data

Local hosts and remote PCs are used to configure and update the system. System update refers to the updating and patching of software, firmware, and operating system components in this context. System configuration comprises configuring or altering the input/output signal binding parameters of RTUs, IEDs, and workstations. This section focuses mostly on remote configuration and update methods. Multiple parties are involved in these two data communications, including software and hardware providers, power companies, and maybe outsourced IT businesses.

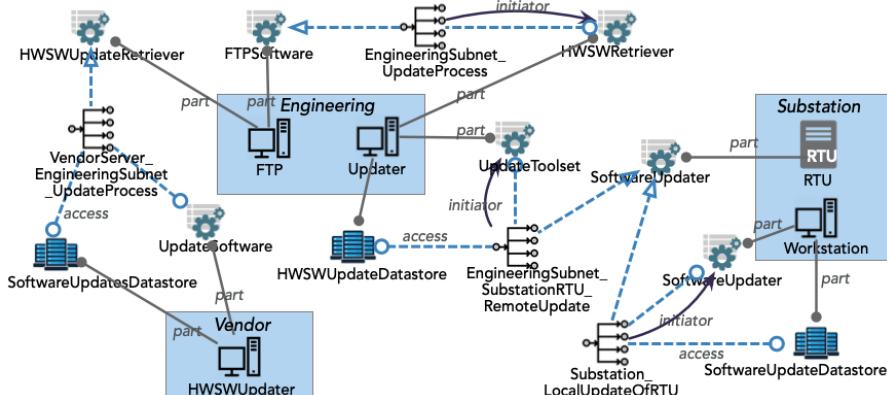


Figure 8.20: System updating data stream

More specifically, Figure 8.20 presents a remote updating process starts with an update request sent out from the *Updater* hosts in *EnergySupplier* network or *Engineering* network, towards the *HWSWUpdater* host in *Vendor* network. Software and firmware updates arrive in the *FTP* server of either *EnergySupplier* network or *Engineering* network, before being transferred to the live systems in the SCADA zone and substation zones. Subsequently, the *Updater* hosts of *Engineering* network may remotely upgrade substations such as *Substation\_RTU*. The *Updater* hosts of *EnergySupplier* network can also remotely upgrade *DER* substations through VPN tunnels. Similarly, the config-

uration server of either the *EnergySupplier* or the *Engineering* network support remote configuration of substation software or operating systems. To decrease the network exposure of directly connected hardware like *IEDs* and *SIS-PLCs*, usually, these devices are maintained or upgraded through maintenance servers and configuration databases embedded in locally connected workstations or *RTUs*.

#### (v) Remote login data

Figure 8.21 depicts a scenario for remote login data in which remote operators in the offices of the smart grid utility gain remote access to SCADA workstations by sending a remote login request from the remote desktop in the *LocalOffice* to the HMI server in the SCADA center. Logging onto servers such as *CitrixServer* from the engineering offices enables remote access to the local workstations. Routers and VPNs are typically used for this purpose. Sniffing malware might discover the office users and passwords, allowing attackers to construct VPNs and get access to the control networks.

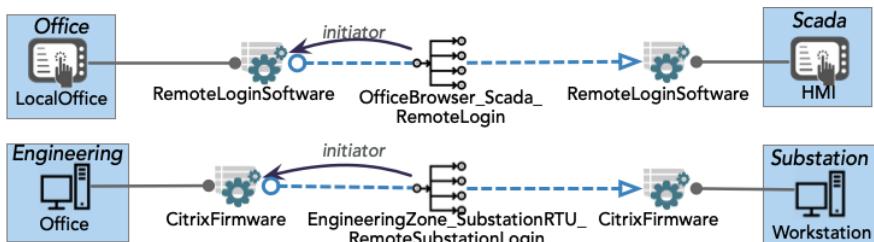


Figure 8.21: Remote login data streams

#### (vi) Web browsing data and mail Data

Figure 8.22 depicts Internet data, which typically consists of online browsing and e-mail communications and is transmitted over local office servers. Incorrect configuration and lack of authorization are potential risks for web servers. Misconfiguration vulnerabilities, such as the existence of unnecessary sample files on a web server, enable attackers to bypass authentication. Weak authentication processes and the lack of a filter for spam and phishing emails are common vulnerabilities of mail servers. For instance, a weak password may allow unauthorized access to a mail server, resulting in the disclosure of sensitive information.

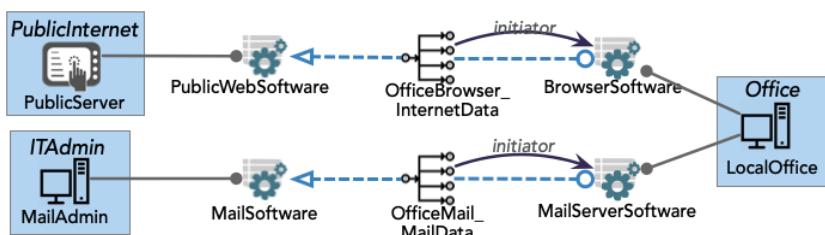


Figure 8.22: Web browsing and mail data streams

## 8.4 STUDY X: APPLICATION AND EVALUATION IN POWER GRID

This study applies the proposed taxonomy, dependence rules and analysis method in two instantiated power-grid models.

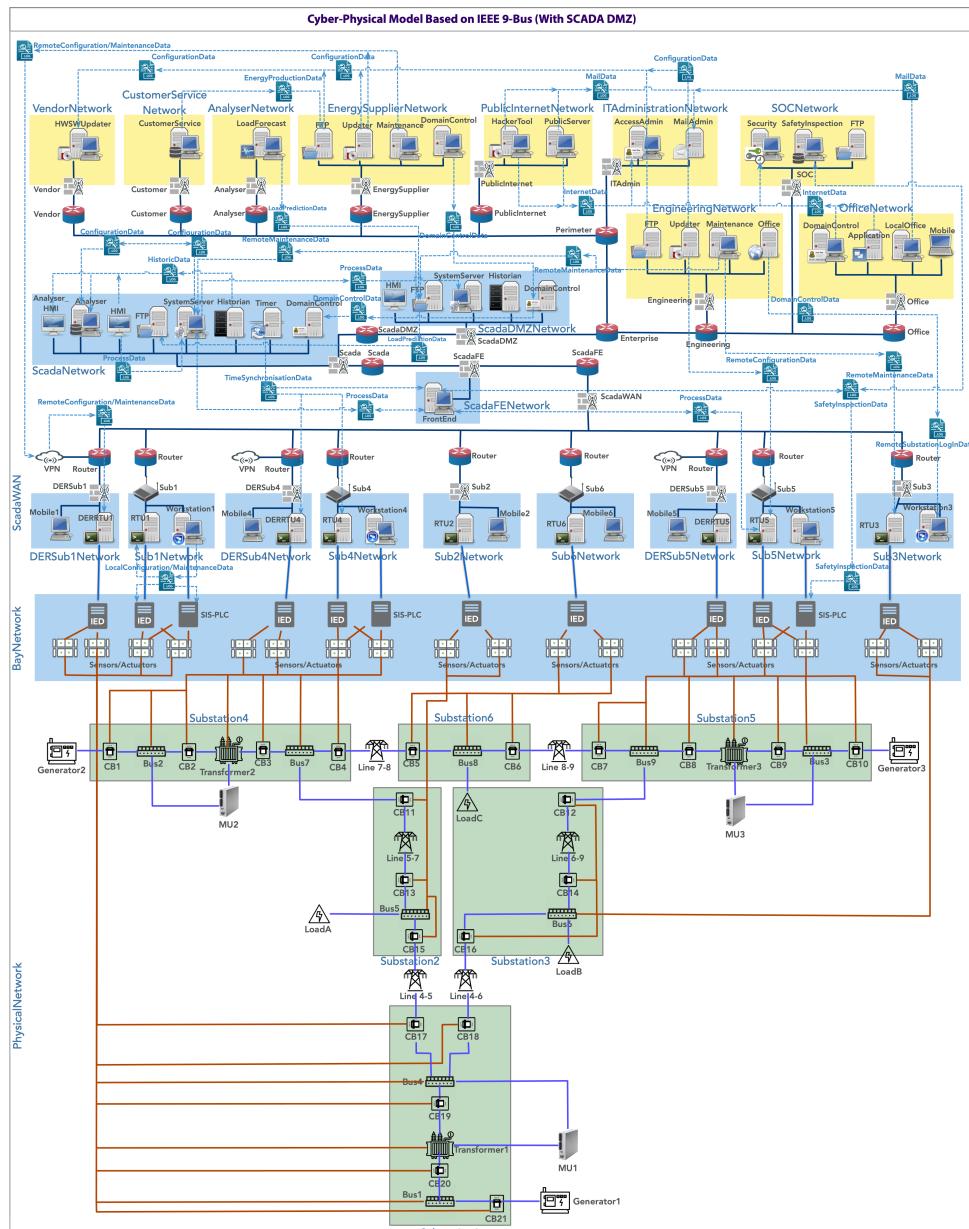


Figure 8.23: Instantiated *Model II* based on IEEE 9Bus (with SCADA DMZ zone) (see Appendix IV for more details)

### 8.4.1 INSTANTIATED POWER-GRID MODELS

We evaluate our taxonomy and rules through two instantiated power grid models *Model I* and *Model II* based on our reference models. These two models differ in terms of whether a SCADA demilitarized zone (or DMZ) is contained or not. The *Model II* example is presented in Figure 8.23 (see more details in Figure 10.6 and Figure 10.7). Details of *Model I* are presented in Figure 10.4 and Figure 10.5 in Appendix IV.

The architectures of these two models are instantiated by following the Purdue model (Williams, 1994) and recommended practices for power-grid security by Homeland Security (2020), to ensure that our models reflect power-grid structure in the real world. For the physical layer of these two models, we follow the IEEE 9-bus system that is commonly used in electricity performance analysis (Sharma, Velgapudi, and Pandey, 2017).

*Model I* consists of 994 components, 1602 topological and functional dependencies, and 172 data flows transferred between network applications. *Model II* contains a SCADA DMZ (Stouffer, Falco, Scarfone, et al., 2011) as a protection layer between IT and OT networks. This DMZ zone contains replicated SCADA servers and historians. A firewall grants the IT network access to the replicated historians. In comparison, *Model II* contains 180 data flows. As depicted in Figure 8.23, we present a simplified explanation of the power system structure that focuses on core features and connections. We highlight some simplified examples for the same type of data flow to illustrate its function and participants.

SCADA WAN contains nine subnets that cover three primary substations (i.e., *Sub1*, *Sub4*, and *Sub5*), three secondary substations (i.e., *Sub2*, *Sub3*, and *Sub6*), and three DER (i.e., *DERSub1*, *DERSub4*, *DERSub5*). Each primary substation subsumes one RTU and one local workstation, while each secondary substation or DER substation subsumes one RTU and one mobile workstation. Here, DER substations and secondary substation are connected to *IEDs*, while primary substations are connected to both *IEDs* and *SIS-PLCs*. Substation automation is achieved through RTUs. For instance, *DERRTU4* is connected with *Bus2*, *CB1* and *CB2* to monitor and control *Generator2*.

The data flows in our instantiated *PG-Model I* mostly follow the examples shown in Figures 8.17 to 8.22.

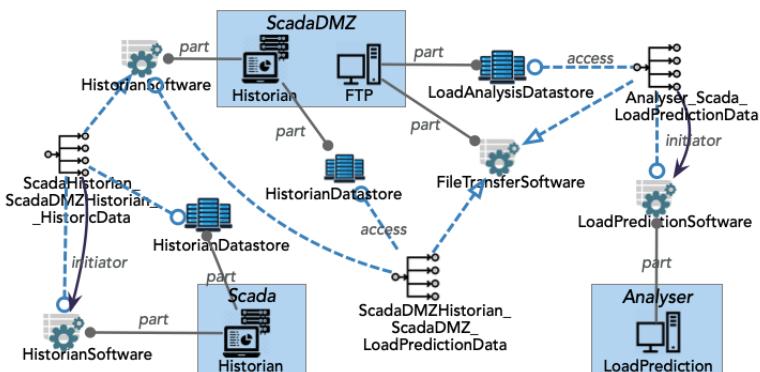


Figure 8.24: Historical analysis and load-prediction data streams in *Model II* (with SCADA DMZ zone)

Some data flows in *PG-Model II* differ. The differences between data flows in *PG Model*

*I* and *PG Model II* are:

- In *PG-Model II*, the historic process data is duplicated, transferred, and stored in a replicated historian in the SCADA DMZ FTP and be accessed to IT network like the *Analyser* network-zone, as illustrated in Figures 8.24. Instead, historic data is accessible from *Office* network-zone in *PG-Model I*, as shown in Figures 8.18.
- In *PG-Model I*, system updating and system configuration are performed both through local hosts and remote computers, as illustrated in Figure 8.20. In comparison, for *PG-Model II*, SCADA technicians request update from hardware and software vendors, and store retrieved data in the FTP server of SCADA DMZ.
- In *PG-Model I*, remote engineers can directly log into local workstations through VPNs, as illustrated in Figures 8.21. In *PG-Model II*, remote operators log into the built-in HMI server in the SCADA DMZ zone to run certain HMI programming software.

#### 8.4.2 MODEL-BASED DEPENDENCE ANALYSIS AND CASCADE MODELING

ConceptBase is an implementation tool that permits the specification of specific relationships using deductive principles. A data link between two components, for instance, is specified once and evaluated as a relation. The data links between cyber components and OT devices such as RTU are of special relevance, as these data streams may be monitored or changed by hostile actors. The query below returns such data streams:

```
DataStreamToRTU in GenericQueryClass isA UnidirectionalDataStream with
parameter
    rtu : RTU
constraint
    cs1 : $ exists comp/Component (~rtu subcomponent comp) and
          (this receiver comp) $
end
```

Using queries allows us to enumerate component dependencies without manually navigating a complex network topology. Instead, we can automate the calculation of dependencies. Based on the criteria stated in the preceding subsection 8.2.1, we programmed our functional dependence rules in ConceptBase. In addition, we implemented comparable ConceptBase queries to extract dependencies of our instantiated models, as shown below:

```
FunctionalDependentOn in QueryClass isA Component with
computed_attribute
    functionallyDependentOn : Component
constraint
    dfc1 : $ (this functionalDependence ~functionallyDependentOn) $
end
```

We query our instantiated power-grid models to determine which components depend on a certain node functionally. Functional dependence is transitive. We identify six components from *Model I* that depend on *Sub1 RTU* functionally. We further extract multiple components that have functional dependence on these six components, which shows transitive functional dependencies. This generates a list of dependent matrices for the targeted model. Such dependence matrices support statistical analysis or graph

modeling using complex network theory. When ranking components with the highest  $N_{i,j}^{FD}$  (introduced earlier in Section 5.2), we observe that *Model I* and *Model II* have the same top-5 components, namely *Scada\_Timer\_TimeUnitSoftware* ( $N^{FD} = 18$ ), *ScadaFE\_FE\_FESoftware* ( $N^{FD} = 10$ ), *Engineering\_Updater\_UpdateToolset* ( $N^{FD} = 6$ ), *Engineering\_Maintenance\_EngineeringToolset* ( $N^{FD} = 6$ ), and *Engineering\_Updater\_FW\_SWUpdateDatastore* ( $N^{FD} = 6$ ).

Figure 8.25 illustrates two scenarios of cascading failures when setting up the same node positions with initial failures.

In the first scenario, we assume that the server *Vendor\_HWSWUpdater* is compromised. In the case of *Model I*, the threat agent may further compromise the *UpdateSoftware* service, upon which false data may be injected to the data receiver like *Scada\_FTP\_SoftwareUpdater*, or leave a backdoor in the host. Furthermore, the threat agent may also alter the data sets in the *SoftwareUpdatesDatastore*. In the case of *Model II*, the threat agent may follow the same attack paths till the *UpdateSoftware* service. Then the threat agent needs to send a data request to the *ScadaDMZ\_FTP\_SoftwareUpdater*, before directly triggering a false-data injection attack.

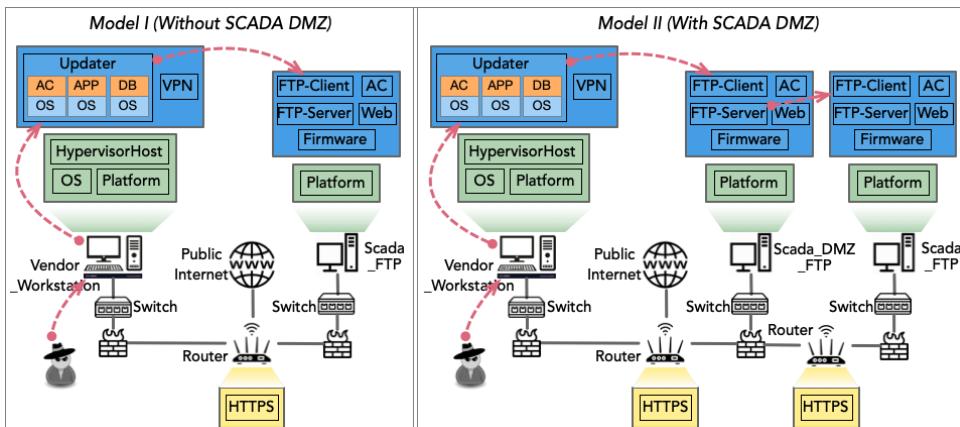


Figure 8.25: Cascade failure analysis in *Model I* and *Model II*

In the second scenario, we assume that *Scada\_Timer\_TimeUnitSoftware* is compromised through the deployment of some existing exploits. Compromising the time unit software in SCADA may allow attackers to view or modify time synchronization data streams between SCADA and controlling substations in both *Model I* and *Model II*. This observation is consistent with our earlier conclusion that nodes with more dependencies may have greater significance in the context of cybersecurity.

In addition, the aforementioned scenarios can use the following rule and query to determine which components may be affected by a failure of the starting node:

```
CascadeFailureRules in Class with
rule
  rule1 : $ forall c1,c2/Component (c1 functionalDependence c2) ==>
          (c2 cascadeFailure c1) $
end
```

```
CascadeFailureNode in QueryClass isA Component with
```

```

computed_attribute
  affects : Component
constraint
  doesaffect : $ (this cascadeFailure_trans ~affects) $
end

```

For large-scale systems such as smart grids, even weak adversaries can trigger cascading failures throughout the entire system, resulting in substantial influence. Enhancing the robustness of system functionalities is one method for enhancing the security of infrastructure systems. Prior to establishing methods for measuring adversarial influence or threat effects, it is crucial to determine the relationships between two components based on their previous contacts. In other words, when modeling and analyzing the vulnerabilities of larger-scale systems, it is essential to take into account the connected impact and composite consequences. Evidence must be compiled from each of the three levels. We support querying both existing system vulnerabilities and possible chained APT vulnerabilities (Chen, Desmet, and Huygens, 2014). One example is illustrated earlier in Figure 8.6.

#### 8.4.3 EXTENDING THE TAXONOMY TO MODEL REAL POWER GRID

In addition to the synthetic investigations *Model I* and *Model II*, we also conducted a research applying the taxonomy to the municipal electricity grid in Sweden. The study's objective was to determine whether the taxonomy could encompass the physical and software components of a real-world power system. The power grid consisted of two larger substations plus more than 200 smaller "transformer" stations serving neighborhoods. The design of the two substations was identical. The transformer stations were available in two variations, one with a single transformer and the other with two.

We developed a network model for the power grid components, including the OT components and control networks. In a second phase, we modeled the control center's software components and, to some extent, the OT components' firmware. We did not model the data flows between the data center and the OT components due to the lack of easily available information. Due to confidentiality restrictions, the real-world study's models were not disclosed and were removed at the study's conclusion.

To summarize, the taxonomy could encompass all elements. In the stage of physical component modeling, two new components types had to be added. One for a specialized balancing equipment used to ensure the neutral power line has a common potential. To mimic circuit breakers with embedded RTU, a second component type was created. Ultimately, we opted to model these integrated devices with the "*subcomponent*" architecture shown in Figure 8.3. The modeling of software components led to similar findings about the taxonomy's scope. We discovered that the control center made extensive use of virtualization; hence, the hypervisor systems needed to be treated as containers of the guest operating systems, which were themselves modeled as containers of the application software.

The study shows that manually modeling complex CIs such as the electrical grid is a time-intensive endeavor. Information about various components of the network model is dispersed among departments and even component manufacturers, making the development of a comprehensive and consistent network model a challenge. Nevertheless, we discovered that most subsystems, such as the transformer stations, had the same architecture. This is also applicable to most configurations of control systems and operations systems. This suggests the usefulness in the development of a duplication function such

as the reference models in sub-section 8.3 so that subsystems could be duplicated fast.

## 8.5 CONCLUSION

This chapter delivers a modeling methodology of intricate CI networks, and related constraints via a taxonomy and instantiated power grid and manufacturing models. These two modules can also serve as a knowledge base of CPS-based CI models that are analyzed by external tools for vulnerability analysis. Static analysis queries are used to pinpoint design weaknesses in the CI layered network. The taxonomy provides a rich set of component and interconnection types to model realistic CI systems. This extends the research on enterprise modeling by covering the physical processes below the technology layer of classical enterprise modeling languages.

We show how the taxonomy can be implemented in a power grid prototype in *Study X*. This study uses two instantiated power-grid models extracted from existing architectures and frameworks like the IEC 62351 series. Such instantiated power-grid models can be leveraged to perform dependence identification and cascading modeling, thus supporting security analysis. We visualize the security topology of the smart grid before and after an attack and answer questions like which IT and OT components are affected by the attack and how the attack propagated throughout the network, in *Study X*. This study also validates the structural, functional adequacy, compatibility, operability, reliability and maintainability of our artifact.

Relations between different classes are categorized as sub-component configuration, data connection, and power connection, and can be easily extended, which shows a high structural strength of our model. We further define the functional dependence rules to formally assess the levels of connections. These dependence rules reuses names from existing reliable ontology and frameworks to ensure conciseness.

Such multi-dimensional functional dependencies also support cascade modeling. The dependence rules are deriving the functional dependence structure from topological and data flow specifications. The rules are fully implemented via the ConceptBase system and efficiently compute the dependencies. The rules can also be used to pinpoint the most “*critical*” components in a CI model in terms of the number of components that depend on them. This extends works on traceability of enterprise models by linking IT components to software artifacts and to the components of the physical system. Our collection of function dependency rules does not include all possible relationships between cyber and cyber-physical layer components. Nevertheless, our dependence rules are multidimensional and serve as a foundation for future expansion.

The reference models can be used in power-grid modeling, presenting high functional adequacy and maintainability. Using the instantiated power grid models provide disciplined and coherent support to specify and group components and coordination mechanisms as a mean to harness the notorious complexity of smart-grid networks. These instantiated models validate the suitability of the CI taxonomy to describe all layers of power-grid systems, from the physical components to the software applications.

We instantiated our model in ConceptBase, which implemented the Telos programming language using an object-like proposition data structure. We established deductive rules and queries for propagating physical power-grid component properties such as nominal voltage and frequency. During the creation and development of our taxonomy, dependency rules, and reference models, we drew from and utilized terminology from existing ontologies and adhered to dependable frameworks such as the Purdue model, NIST SP

800-82, and the IEC 62351 series to strengthen the compatibility of our works.

Part of our taxonomy has been utilized in (and still part of) our master's program, particularly in network modeling and analysis lab assignments of complex CPSs, which indicates operability. The presentation of our proposed taxonomy and reference model uses ConceptBase for its ability to represent both classes and objects in the same database. This allows us to use the taxonomy as constructs of a domain-specific modeling language to represent sample smart grids to any degree of detail. In doing so, we can extend the taxonomy at any time.

# STREAMLINING VULNERABILITY ANALYSIS FOR CRITICAL INFRASTRUCTURES

BY [REDACTED] | PUBLISHED ON [REDACTED]

INTRODUCTION

This report provides an overview of the current state of vulnerability analysis for critical infrastructures and outlines a framework for streamlining this process.

The report begins by defining critical infrastructure and identifying the key challenges faced by organizations in conducting vulnerability analysis.

It then explores various tools and techniques used in vulnerability analysis, including automated scanning, manual penetration testing, and threat modeling.

The report also highlights the importance of collaboration between different stakeholders, such as government agencies, industry associations, and cybersecurity experts, to improve the effectiveness of vulnerability analysis.

Finally, the report concludes with recommendations for improving the efficiency and effectiveness of vulnerability analysis for critical infrastructures.

DEFINITION OF CRITICAL INFRASTRUCTURE

For the purposes of this report, critical infrastructure is defined as any system or asset that is essential to the functioning of society and whose disruption, degradation, or destruction would have a significant impact on public health, safety, or security.

CHALLENGES IN VULNERABILITY ANALYSIS FOR CRITICAL INFRASTRUCTURES

Conducting vulnerability analysis for critical infrastructures presents several unique challenges, including:

- Complexity: Critical infrastructure systems are often highly complex, with many interconnected components and dependencies.

- Scalability: The size and scope of critical infrastructure systems can make it difficult to identify and prioritize vulnerabilities across all assets.

- Regulatory requirements: Different industries and regions have varying regulations and standards for cybersecurity, which can complicate the analysis process.

- Resource constraints: Many organizations managing critical infrastructure have limited resources and personnel dedicated to cybersecurity.

TOOLS AND TECHNIQUES FOR VULNERABILITY ANALYSIS

There are several tools and techniques available for conducting vulnerability analysis, including:

- Automated scanning tools: These tools use pre-defined rules and patterns to identify potential vulnerabilities in network and system configurations.

- Manual penetration testing: This involves manually attacking a system to identify weaknesses and exploit them.

- Threat modeling: This involves identifying potential threats to a system and determining the likelihood and impact of each threat.

COLLABORATION AND PARTNERSHIPS

Collaboration and partnerships are crucial for improving the effectiveness of vulnerability analysis for critical infrastructures.

ACKNOWLEDGEMENTS

Special thanks to [REDACTED] for their contributions to this report.



## CHAPTER 9

# STREAMLINING VULNERABILITY ANALYSIS FOR CRITICAL INFRASTRUCTURES

This chapter answers the research question of how to assess the vulnerabilities of complex CIs with the support of our proposed curated vulnerability database (presented in Chapter 6), the embedded vulnerability-analysis ML algorithms (presented in Chapter 7), and the CI vulnerability taxonomy (presented in Chapter 8). This question can be divided into sub-questions: (i) how to identify vulnerabilities for CIs? and (ii) how to model and assess system-wide vulnerabilities?

In response to these two questions, we propose a vulnerability-analysis orchestration method to connect the proposed artifacts, including the CI vulnerability taxonomy, the cross-linked vulnerability database and the ML-based vulnerability analysis approaches, to identify and assess matched vulnerabilities. This is the core coordinator agent that orchestrates the communication and exchange of data across our suggested modules.

## 9.1 ARTIFACT IV: VULNERABILITY ANALYSIS ORCHESTRATION METHOD FOR CRITICAL INFRASTRUCTURES

This thesis proposes a vulnerability analysis orchestration method that facilitates communication and information exchange between our proposed taxonomy and previously established modules (i.e., correlated database and machine learning vulnerability analysis algorithms) and orchestrates their activities. Figure 9.1 illustrates this vulnerability analysis orchestration method. *Orchestration* refers to the process of integrating diverse cybersecurity modules in order to simplify the interaction between vulnerability data and CI multi-layer network in order to provide a higher level of automation in vulnerability assessment. CI models created using our CI taxonomy provide the query generator with organized system configuration data. This query generator delivers tags in the *CPE* metadata format, from which cybersecurity professionals may pick and query the localized database. This database of vulnerabilities cross-links several open-source vulnerability repositories and enumerations and returns a list of CI vulnerabilities based on the query. ML models trained on historical vulnerabilities are then employed to assign missing severity, threat, and higher abstraction level *CWE* labels to these CI vulnerability cases. Finally, these instances with new security indications are returned to the CI taxonomy to provide additional visualization and static query-based analysis.

More specifically, Figure 9.2 presents a model-based security engineering method in which the proposed taxonomy plays a vital role to facilitate dependence analysis and vulnerability assessment. CI models instantiated via our taxonomy give organized system configuration information to support vulnerability queries. Next, vulnerability analy-

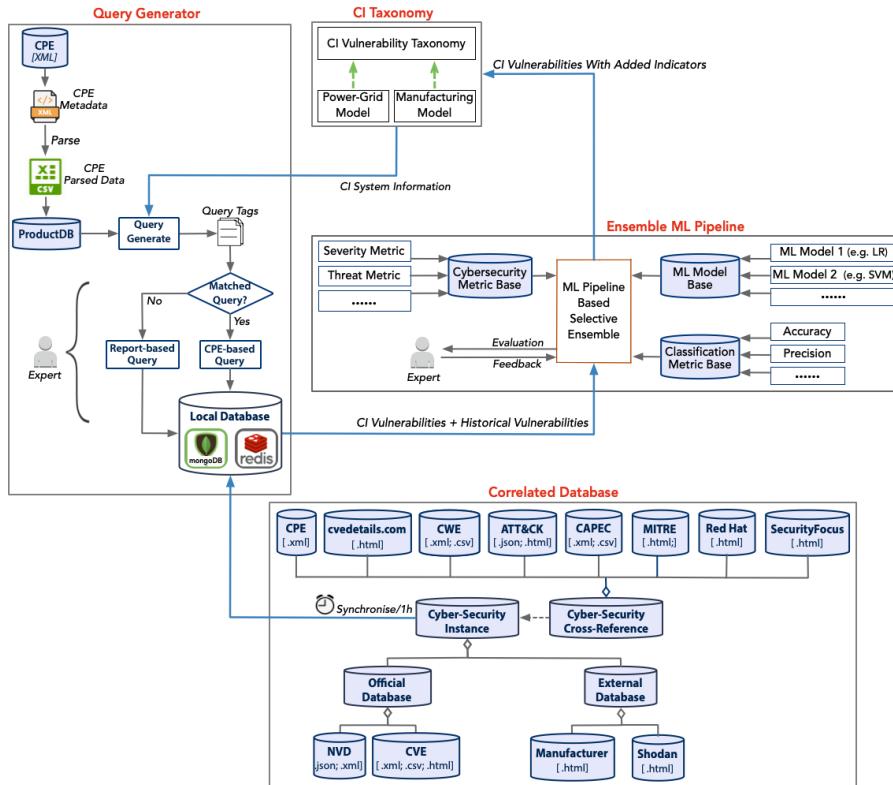


Figure 9.1: Orchestration of the proposed artifacts and instantiations

sis techniques are utilized to assign severity, threat, and vulnerability labels to these CI vulnerability situations. Furthermore, these vulnerability cases with added security indicators are inserted into the existing CI models to facilitate further visualization and static query-based analysis.

An attacker may materialize an APT utilizing a vulnerability chain, in which vulnerabilities on one set of components grant access to attacks on another set of components. Our taxonomy facilitates modeling and detection of such vulnerability chains, with one example illustrated in Figure 9.3.

$V-x$  in Figure 9.3 relates to several vulnerabilities. Additionally, we utilize  $A-x$  to denote an advanced assault stage. An attacker might use the default password configuration ( $V-1$ ) of a design engineer's account to compromise this account ( $A-1$ ), and then attempt to infiltrate an associated designer workstation using compromised accounts in a subsequent attack ( $A-2$ ). If the designer workstation has insufficient authentication management ( $V-2$ ), then this subsequent assault ( $A-2$ ) may have a greater chance of success. In addition, the threat agent may initiate a second attack ( $A-3$ ) to acquire access to a database server through a compromised designer workstation. A potential vulnerability of weak access control ( $V-3$ ) could permit an attack ( $A-3$ ), which could allow a threat agent to launch a second attack ( $A-4$ ) to manipulate certain geometry CAM programs in the control server if there is a further vulnerability of an unencrypted database ( $V-4$ ). Due to poor communication between CAM-engineers and security officers ( $V-5$ ),

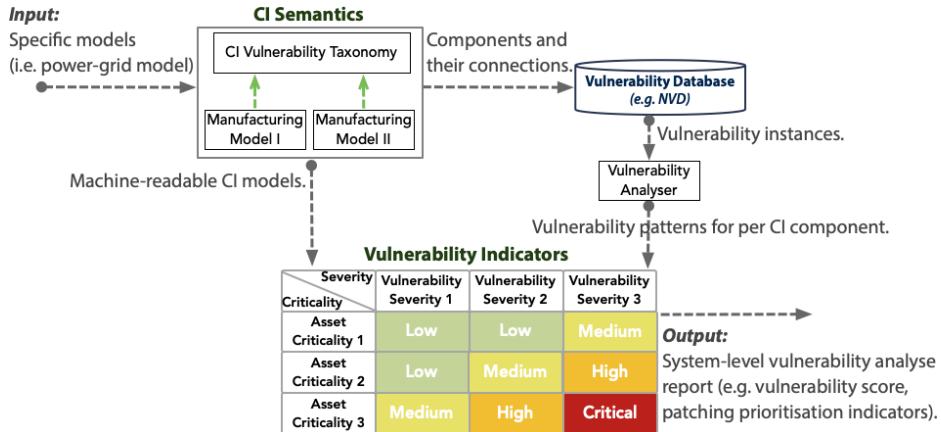


Figure 9.2: Connection between the proposed taxonomy and other artifacts

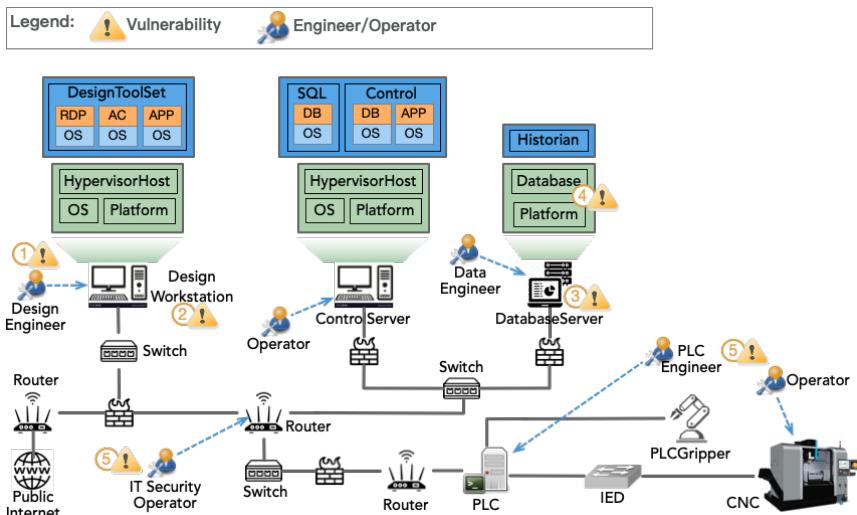


Figure 9.3: A scenario of vulnerability chain in manufacturing system

previously successful CAM program code modification attacks may go undetected. IT-personnel, OT-personnel, and local operators may experience communication gaps if the IT department and OT department are located in separate locations. In this scenario, the altered CAM file may reach the PLC uncorrected, causing NC or CNC machines to move in the incorrect direction. Due to communication inadequacies, this might have serious consequences for the production process.

The aforementioned chained vulnerabilities can be defined with security professionals' expertise, or identified by following the connections between *CWE*, *CAPEC* and *ATT&CK* attributes of investigated vulnerability instances. The vulnerability chain example shown in Figure 9.3 can be modeled using the proposed taxonomy, as presented in Figure 9.4. It is interesting that *CAPEC-560* can follow both *CAPEC-49* and *CAPEC-70*, indicating multi-step attack graph following the vulnerability chain. Identification

of such vulnerability chains is valuable for system-level vulnerability assessment.

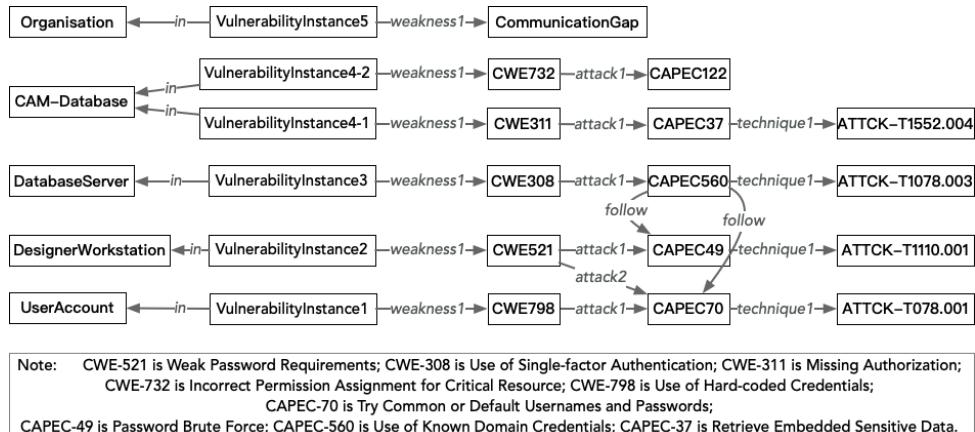


Figure 9.4: Modeling and identification of vulnerability chain

## 9.2 STUDY XI: APPLICATION AND EVALUATION IN MANUFACTURING

We interviewed 2 industrial-production professionals and 2 operators of a manufacturing company to collect information about manufacturing networks structure, to set up models for the manufacturing systems. We then instantiated a human robot collaborative (HRC) assembly system to validate our proposed approach.

### 9.2.1 PROTOTYPE OF MANUFACTURING SYSTEM

The manufacturing model adopts similar structure of the power grid models, and is consisted of physical-, control-, and cyber networks, as illustrated in Figure 9.5. Human roles are also marked in this diagram to show how can manufacturing engineers and cybersecurity engineers collaborate for vulnerability identification, assessment and management. One of such examples is already presented earlier in Figure 8.7.

The physical layer incorporates manufacturing machinery such as CNC, which is used to automate machining tools, PLC gripper system where the gripper controller PLC is placed, Robot system where the robot and conveyor are located, conveyor for routing material through machineries, as well as distributed sensors and a variety of cameras for capturing production-related measures. On HMI embedded devices, local operators can see synchronized instructions and validate their status. Meanwhile, application engineers do routine maintenance on these workstations.

The control layer also includes networks for communications between CNC, PLC, Robot, and other equipment, often utilizing a master-slave structure. Typically, a master device (e.g., the CNC) starts inquiries and delivers them to a slave device. Through a local communication network such as Modbus, this slave device (e.g., Robot PLC) receives the inquiry and transmits the needed data to complete the transaction. Note that CNC and PLC controllers are connected to the actual equipment but not the control network. In

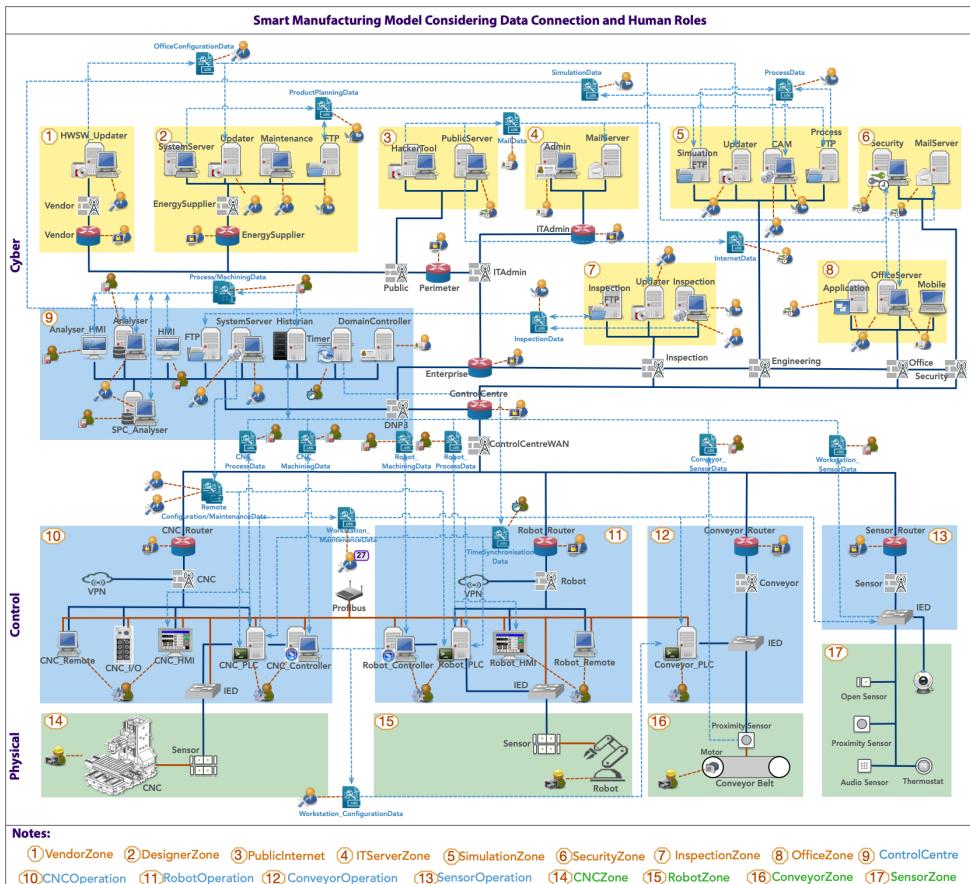


Figure 9.5: Manufacturing Network  
(Figure reproduced from Jiang et al. (2019))

the integrated HMIs of the control center, human employees may directly monitor assembly tasks. In order to do this, computers in the CNC are designed to gather data from the physical production system and display it to the human operator, enabling CNC and Robot operators to perform remote machining and supervise the process. In addition, the control center typically includes a historian server for displaying historical data, an application server that retrieves data from the historian for analysis and also supports other software applications, and a domain controller for implementing security policies such as access control. There are also additional user jobs in the control layer, such as PLC programmers who develop field device operational instructions for efficient production flow.

The cyber layer of manufacturing system share similar functional zones like the cyber network in power grid systems. For instance, *DesignerZone*, *SimulationZone* and *InspectionZone* host servers that support machining model design, simulation and production inspection, separately. Generally, CAD designers in the *DesignerZone* create CAD files and transfer them as product planning data to the *SimulationZone*. CAM engineers receive the CAD files and conduct simulations, and further creates G-code or

M-code files (i.e., programming instructions that tell machines what to do) out of the ProcessData. Data engineers query the simulation datastore from the *ControlCentre*, and then further divide queried process and machining data into CNC machining data process data to CNC machines for production purpose, as well as robot machining data and process data for process-control purposes. Meanwhile, process and machining data from the workstation are sent back by CNC PLC and controller separately, under the request of data operators in *ControlCentre*. These data would be under quality inspection by production engineers and statistic process control (SPC) data engineers. Additionally, IT administrators and domain operators in the *ITAdminZone* are in charge of security maintenance including administrative tasks and authorization management such as mail service and user management.

### 9.2.2 HUMAN ROBOT COLLABORATIVE ASSEMBLY SYSTEM

In our manufacturing study, a HRC assembly system was used. A HRC system is a collaborative environment in which people and robots perform concurrent activities (Wang et al., 2019). Given that human operators are directly involved in the assembly system and work side by side with robots, the significance of cybersecurity protection and maintenance of such HRC systems cannot be understated. Malicious alterations to robot movement orders may lead to the damage of human employees if they are not properly monitored.

Due to the fact that constructing the HRC assembly system is not a contribution of this thesis, we briefly describe its primary structure and linkages that are significant to cybersecurity analysis. The *Symbiotic Human-Robot Collaborative Assembly: Technologies, Innovations and Competitiveness* (2022) (SYMBIO-TIC) project offers further information on the HRC system. The third demonstration that is located at ASSAR Industrial Innovation Arena in Sweden is particularly relevant. The information that serves this case study is collected through three main manners: (i) outputs of SYMBIO-TIC project, including published papers and reports; (ii) field study in the actual HRC settings in ASSAR; and (iii) interviews with previous project members of SYMBIO-TIC.

This HRC assembly system consists of three work stations, one tool changing station, a robot (*ABB IRB 2600-20(12)/1.65*) with a PLC gripper, and a conveyor. Operators mostly cooperate with the robot at the three work stations. The cameras capture their position and motions. (products of *Microsoft Kinects*) for assembly process planning and scheduling purposes. To do so, several systems are employed to streamline the data flows between these working stations and robots, namely an *UnitController* system for assembly data analysis and sending commands, a *Cockpit* system for assembly process planning, a *CollisionAvoidance* system for human-robot movements analysis, and a *WorkerIdentification* system to recognize workers' movements, as illustrated in Figure 9.6.

More specifically, the *WorkerIdentification* and *UnitController* systems assess workers' and robot's locations and their availabilities, separately. Based on the information of available workers and robot, the *Cockpit* system completes assembly process planning and scheduling for a batch of given product, and further sends these assembly operations to the *UnitController*. *UnitController* requests detailed information for each assembly operation (e.g., a robot's movement from one point to another) that is encapsulated in function blocks by graphical robot programming software like *Drag&Bot* (2022). Such gripper opening and closing instructions are provided by robot simulation software like *RobotStudio* (2022). The *UnitController* translates function blocks into running codes and transfers them to the robot controller (e.g., *IRC5* (2022) controller) to control the

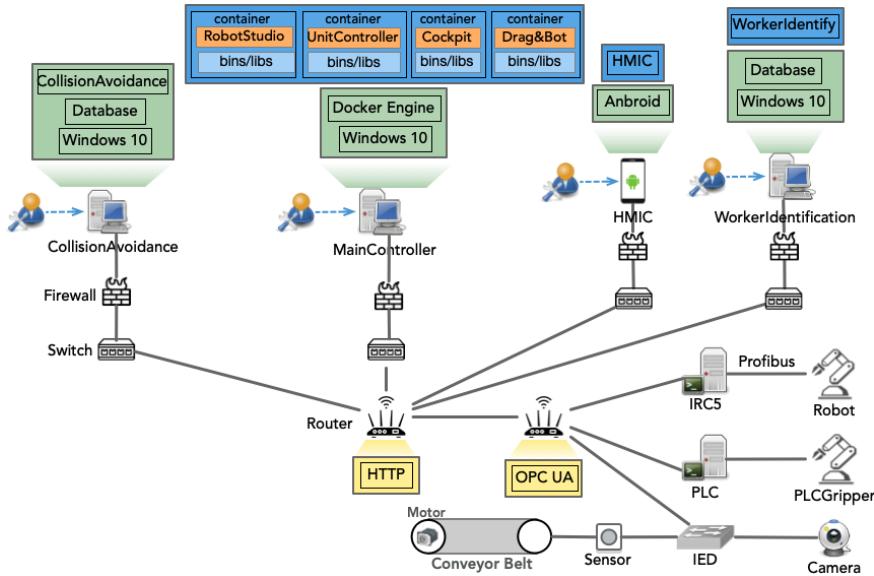


Figure 9.6: Human robot collaboration system that integrates *SYMBIO-TIC*

robot, and into I/O (referring to input and output) signals to PLC to control the behavior of the gripper. Meanwhile, *UnitController* sends the instructions for workers and working stations to the *HMIC* (referring to human machine interface controller) which are usually embedded in a tablet or a smart phone. Workers can then enter operation execution status into the *HMIC*. Considering the goal of higher resilience of the HRC assembly system, duplicated assembly process data are stored in the *UnitController*, the *CollisionAvoidance* and the *Cockpit* system, while synchronised through timers. When the *CollisionAvoidance* detects potential collisions between human and robot, it revises the current trajectories of robot and sends instructions back to *UnitController* to avoid collision.

The aforementioned simulation and programming software packages, including *Robot-Studio*, *Drag&Bot*, *UnitController* and *Cockpit*, are embedded in one workstation that uses *Windows 10* for x64-based system, as illustrated in Figure 9.6. The *CollisionAvoidance* system is embedded in another workstation also running on *Windows 10*. Meanwhile, the *WorkerIdentification* software is embedded in another computer running on *Windows 10*. *HMIC* software package is embedded in one mobile phone and one pad, both of which are running on *Android* operating system. Among all these developed software components, only *Drag&Bot* allows remote access directly. The other software components like *Cockpit* are each separately encapsulated in software containers like *Docker* (2022), to ensure each running software perceiving the machines in a secure way with lower chance of data leakage. Communication between processes and memory allocation is strictly controlled and isolated. Access control policy is well embedded by assigning roles clearly. For example, *Cockpit* user can only log into the *Cockpit* system, but does not have access permissions to other software, configurations, libraries or databases.

All machines and services are operating on a single, password-protected local network. A router from *ASUS* is used to encrypt the network using a typical SSH (referring to

Table 9.1: Vulnerability patch decision making considering criticality and severity

Component	Criticality	Number of Vulnerability	Average Severity
<i>HRC_MainController_UnitController</i>	6	N/A	N/A
<i>HRC_MainController_RobotStudio</i>	1	1	7.4
<i>HRC_MainController_Cockpit</i>	2	N/A	N/A
<i>HRC_MainController_Drag&amp;Bot</i>	2	N/A	N/A
<i>HRC_MainController_DockerEngine</i>	4	8	7.23
<i>HRC_MainController_OperatingSystem</i>	5	17	7.5
<i>HRC_WorkerIdentification_OperatingSystem</i>	1	17	7.5
<i>HRC_WorkerIdentification_WorkerIdentification</i>	1	N/A	N/A
<i>HRC_CollisionAvoidance_OperatingSystem</i>	2	17	7.5
<i>HRC_CollisionAvoidance_CollisionAvoidance</i>	2	N/A	N/A
<i>HRC_HMIC_OperatingSystem</i>	1	2	7.3
<i>HRC_Router</i>	5	4	8.08
<i>HRC_IRC5</i>	2	2	9.8
<i>HRC_PLC</i>	2	5	7.2

secure shell) encryption layer. The hub features IP-reservable software firewalls. The hub links certain computers to specific IP addresses. The software is set to anticipate communication and connects to certain services.

### 9.2.3 MODEL-BASED VULNERABILITY ASSESSMENT FOR HUMAN ROBOT COLLABORATIVE ASSEMBLY SYSTEM

Our criticality study consists of two steps: (i) a static analysis utilizing the taxonomy, and (ii) a conversation with stakeholders to determine the weighting of criticality. We evaluate the critical assets of the HRC assembly system in accordance with the reliance criteria outlined in sub-section and rank the HRC system components from most critical to least critical. Here, the criticality of a component is determined by the number of other components that depend on it functionally. The five most critical components are *HRC\_MainController\_UnitController* ( $N^{FD} = 6$ ), *HRC\_Router* ( $N^{FD} = 5$ ), *HRC\_MainController\_OperatingSystem* ( $N^{FD} = 5$ ), *HRC\_MainController\_DockerEngine* ( $N^{FD} = 4$ ), and *HRC\_MainController\_Drag&Bot* ( $N^{FD} = 2$ ). We then discussed with the project members of SYMBIO-TIC to validate these identified critical assets and to weight their criticality. Besides the five most critical components, these project members also suggest the importance of the physical PLC gripper and robot, and also data assets stored and communicated between *RobotStudio*, *Drag&Bot*, *Unit-Controller* and *Cockpit*.

We gathered system configuration and component data for the HRC assembly system, and then generated queries for our localized vulnerability database. Note that we only gathered 14 essential components for the functions depicted in Figure 9.6. By querying the database, we obtained 39 vulnerability instances that are published before November 3, 2021, which is composed of 3 critical, 30 high, and 6 medium severity vulnerabilities.

Table 9.1 displays the criticality level when functional dependencies, the number of discovered vulnerabilities, and the average severity score for these vulnerabilities are considered.

In the APP layer, Docker containers bundle program code and dependencies. A container is reasonably separated from other containers and its host system. Therefore, databases are not shared between computers. Nonetheless, several known Docker vulnerabilities, such as the container breakout vulnerability, allow an attacker to further exploit confined software through a backdoor. Table 9.1 suggests that operating system of the *MainController* can be given the highest prioritization.

During our model-based evaluation, we also identified a structural weakness in the HRC system. Even though all databases (such as *Cockpit*, *HMIC*, and *Drag&Bot* databases) are password protected, passwords are saved in plaintext in configuration files. For instance, the configuration file stores the *RobotStudio* password to enable data connection with *Drag&Bot*. This vulnerability is classified as *CWE-260* and may allow an attacker to obtain privileges or assume identity. Once adversaries get access of the *RobotStudio* system, they may alter robot production procedures and do damage to the entire HRC system.

We also learned that this static analysis only covers a subset of the system's components. And hence, some vulnerabilities may be left out. Still, the suggested taxonomy and instantiated models allow for further development with more complex systems and elucidated rules for query-based vulnerability analysis.

### 9.3 CONCLUSION

A vulnerability-analysis orchestration method presented in this chapter contributes to data-driven vulnerability analysis for critical manufacturing by correlating subtle modules for vulnerability data curation, vulnerability feature allocation, and system-wide manufacturing analysis, which are used to support CI vulnerability assessment.

*Study VII* shows the effectiveness of our method in terms of consistent search and query support for relevant vulnerabilities of a targeted component, as well as knowledge reuse of CPS-based CI configuration. The CI taxonomy presented in Section 8.1 was applied to generate reference models of manufacturing and further deliver instantiations for an actual HRC assembly system, which demonstrates good functional adequacy and maintainability. The localized database and query generator support the retrieval of individual vulnerabilities. In addition, the instantiated HRC assembly model supports model-based vulnerability assessment, which assists in the identification of structural vulnerabilities. With the dependence rules, we further model chained vulnerabilities to support system-level vulnerability assessment, in which multiple vulnerability instances need to be assessed together. In this way, we evaluate both individual and structural vulnerabilities to help with the prioritization of vulnerability patches.

The presented study shows the strength of our method in terms of consistent search and query support, as well as knowledge reuse extracted from vulnerability repositories and CI system configurations. The proposed vulnerability analysis orchestration method helps a system owner identify the most critical components for cybersecurity protection before a patch is available. Components containing the same vulnerabilities require different levels of patch prioritization, as some components are protected or less accessible (e.g., an isolated device). The proposed taxonomy and implemented queries support prioritization analysis that considers the system architecture. This taxonomy can also be easily extended to support analysis of vulnerability chains with higher complexity.







# CHAPTER 10

## CONCLUSION

This final chapter discusses how the main contributions of this thesis address the challenges outlined in Section 1.2, meet the objectives presented in Section 1.3 and answer the research questions proposed in Section 1.4. This chapter ends with a suggestion of some preliminary users of this study, as well as possible expansions and future directions that would build on the presented research.

### 10.1 REVISITING THE OBJECTIVES AND THE RESEARCH QUESTIONS

This section revisits the objectives and research questions presented earlier in Section 1.3 and 1.4, addressed with the proposed answers while highlighting contributions.

The work described in this thesis engaged in understanding and suggesting solutions to bridge the knowledge gap among several essential functional modules in the current CI vulnerability assessment and management processes. In doing so, it addresses four main challenges: (i) heterogeneous and diverse vulnerability data sources; (ii) subjective and human-centered analysis process; (iii) CI dependencies that exacerbate analysis complexity, and (iv) gaps in CI vulnerability management. To tackle these challenges, four research questions were formulated, as illustrated in Figure 10.1 and further discussed next.

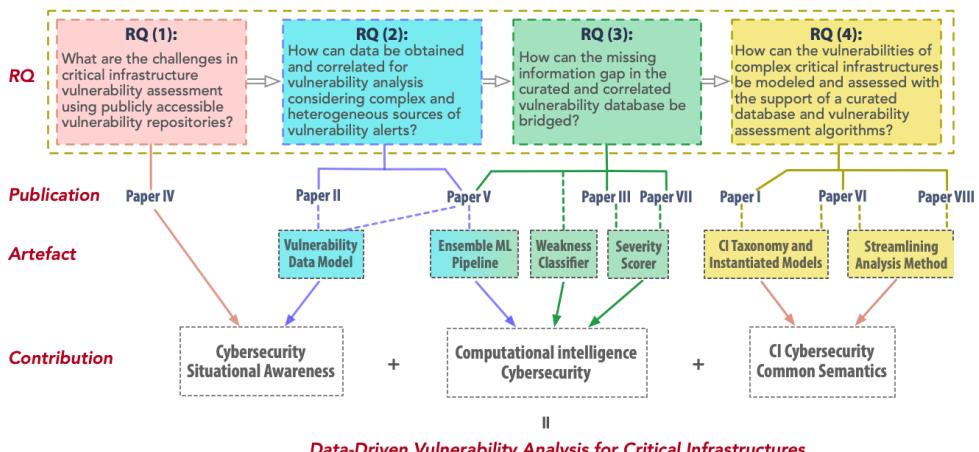


Figure 10.1: An overview of relationships between artifacts and research questions

*RQ (1) - What are the challenges in critical infrastructure vulnerability assessment using publicly accessible vulnerability repositories?*

Exploratory experiments described in *Paper IV* and Chapter 5 facilitate identification of research gaps and assignment of problems. Specifically, vulnerabilities are disclosed using disparate standards and vocabularies, which complicates their understanding and sharing. We learned through our baseline *Study I* that *NVD* is viewed on the website or downloaded in JSON or XML format. On the other hand, *CWE* is downloaded in XML, HTML, or CSV format. Even when both are downloaded in XML format, their syntax and semantics remain distinct. In the meanwhile, vulnerability records from the past reveal a pattern of variation in the deployment of security measures. For instance, *NVD* analysts employ a collection of 241 unique *CWE*-IDs to manually assign weakness labels to vulnerability occurrences. This exploratory baseline research underlines the heterogeneity and complexity of vulnerability data sources accessible to the public. Additionally, some vulnerability instances are not allocated with weakness and threat labels or severity scales. For example, more than 29 % vulnerabilities have no assigned *CWE* weakness labels. Then, the CI stakeholder survey (*Study II*) and *Study III* identify other problems with these vulnerability repositories, including synonyms, incompleteness, and inconsistency. *Study III* also evaluates the impact of data inconsistencies in vulnerability repositories on vulnerability assessment of actual IT and OT systems. These three exploratory investigations demonstrate the requirement for a cross-linked vulnerability database as well as data cleaning and fusing strategies to appropriately preprocess vulnerability data.

*RQ (2) - How can data be obtained and correlated for vulnerability analysis considering complex and heterogeneous sources of vulnerability alerts?*

In response to this question, a common data model (CDM) is provided along with correlation and query generating techniques for integrating diverse data sources into semantic categories based on their co-occurrence, as presented in Chapter 6. On the basis of this unified semantic model, vulnerability data collection and subsequent correlation opportunities are combined with a new query generation logic, to serve as the foundation for context-specific vulnerability assessment. We further instantiated a correlated and synchronized database to facilitate the retrieval and analysis of vulnerabilities. Thus, security operators at various SOC levels are equipped with a customized and synchronized database that retrieves vulnerability data from several internet sources. *Study IV* shows that such linkage and aggregation of vulnerability data sources provides more in-depth analytics that link relevant exploits and threat-agent activities with the evaluation of general detection and defensive measures. The process of consolidation and normalizing allows for the incorporation of numerous, changing formats, while also enhancing current data, as shown in a comparative analysis *Study V*. A vulnerability intelligence system should seamlessly link freshly gathered data to old data and allow analysts to execute more complicated data bindings. Specifically, vulnerability instances are gathered synchronously from repositories to aid operators in assessing current vulnerability trends and to further reduce the risk-window caused by found vulnerabilities, similar to how the localized database in *Paper II* is utilized.

*RQ (3) - How can the missing information gap in the curated and correlated vulnerability database be bridged?*

Using correlation approaches presented in Chapter 6, data-driven security indicators generate a correlated database from several vulnerability-alert repositories and standard enumerations. A knowledge-level layer then passes the vulnerability data to a rule-based classifier, which correlates them into patterns. To answer *RQ (3)*, an optimization approach is developed to select the ideal machine-learning base algorithm(s) for generat-

ing effective ensemble models for vulnerability analysis and threat modeling, based on a standard set of machine-learning performance criteria. ML eliminates manual, time-consuming report analysis and reduces the time security analysts spend analyzing large volumes of security alerts related attributes, such as the automatic vulnerability *CVSS* severity scoring system proposed in *Paper III* and improved in *Paper VII*, as well as the *CWE* weakness classifier presented in Section 7.4. These two ML classifiers show better performances when compared to related works in experimental studies, *Study VI* and *Study VIII*. These ML classifiers show usefulness and applicability in vulnerability analysis automation in *Study VII*. Furthermore, using the ensemble method presented in *Paper V*, newly discovered vulnerability instances are given to a ML pipeline that classifies these instances based on threat-, vulnerability-, and weakness-patterns. *Study IX* investigates different ensembles of five supervised ML models (i.e., LR, NBSVM, LSTM-RNN, MLP, and KNN), which reveal better performances than their individual performances.

*RQ (4) - How can the vulnerabilities of complex critical infrastructures be modeled and assessed with the support of a curated database and vulnerability assessment algorithms?*

To answer this question, we suggest a modeling methodology of intricate CI networks, and related constraints via a taxonomy and instantiated reference models for power grids and manufacturing. These two modules can also serve as a knowledge base of IT/OT convergent CI models that are analyzed by external tools for vulnerability analysis. Current CI vulnerability management is challenging due to the knowledge gap between IT security and OT security, and also different terminologies used in these two domains. Our model bridges such gaps with common semantics, and supports query of vulnerabilities across the CI layers. Static analysis queries are used to pinpoint design weaknesses in the CI layered network. Using the instantiated CI reference models provides disciplined and coherent support to specify and group components and coordination mechanisms as a mean to harness the notorious complexity of smart-grid networks. This taxonomy supports reusable and efficient usage through CI model instantiations with standardized virtual replica for cyber connections and cyber-physical processes. Our research contributes to the cybersecurity analysis of CIs through a taxonomy that is a model of a domain describing objects that inhabit it. This taxonomy is a system of categories of IT, OT, and physical objects that was derived practically or conceptually. Thus, it facilitates the comprehension and organization of the knowledge of the broad field of CI cybersecurity, as presented in *Paper I*, *Paper VI* and *Paper VIII*.

To summarize, three objectives were formulated to guide the studies in this thesis. Namely:

*Objectives:*

- (A) To expand the scope of knowledge about vulnerability alerts and curate meaningful context of vulnerability analytical processes.
- (B) To assess identified vulnerabilities with enhanced levels of automation that reduce existing information gaps induced by ad-hoc and subjective auditing processes.
- (C) To model vulnerability in CIs in a reproducible manner that supports vulnerability assessment and increases the level of security awareness.

This thesis includes exploratory studies of existing vulnerability repositories and CI structures, which leads to the development of a correlated vulnerability database and a CI

taxonomy with clearly defined relationships between CI component, security attributes such as vulnerability, and security stakeholders, addressing *Objective-A*. Answers to *Objective-B* are offered by utilizing the cross-linked vulnerability repository and ML techniques to automatically assign labels of vulnerability attributes, and are further strengthened by the proposed vulnerability-analysis orchestration system and its embedded workflows. This framework also satisfies *Objective-C*, together with the CI taxonomy and instantiated models for the power grid and industrial manufacturing.

This thesis contributes to data driven vulnerability analysis with a set of proposed models and methods, namely vulnerability data model (*Artifact I*), ML-based analysis method (*Artifact II*), CI vulnerability taxonomy and dependence rules (*Artifact III*), as well as vulnerability-analysis streamlining method (*Artifact IV*). These artifacts are validated in a series of studies (see Table 4.2) to evaluate their utility, following standardized metrics drawn from literature review. Together, this thesis (i) identifies challenges of using open-source vulnerability repositories and scoring systems; (ii) expands vulnerability knowledge scope with data model and correlation method; (iii) fills the information gaps in vulnerability repository with ML methods that also increase the level of automation in vulnerability assessment; (iv) detects dependencies and vulnerabilities through CI vulnerability semantics; and (v) bridges the gaps among different security functions and stakeholders.

Based on the results presented in this research study and summarized in the previous paragraphs, it can be concluded that the research questions have been answered and the research aim has been filled that is to investigate workflows that bridge the knowledge gaps among different security functions, such as vulnerability management, report analysis, and infrastructure networks monitoring, in order to correlate vulnerability findings and coordinate mitigation responses in complex CIs.

## 10.2 FUTURE WORKS

This thesis contributes to the development of data-driven solutions for CI security, particularly in the areas of vulnerability assessment and management for complex CI systems considering heterogeneous and different security alerts. Multiple studies are conducted to demonstrate the validity of the given solutions. Nevertheless, additional research and inquiry into alternate research prospects are still required.

### (i) Data reliability validation automation:

To further improve public vulnerability data data sources, we intend to integrate additional sources of vulnerability data in order to incorporate the opinions of a wider range of stakeholders. We also plan to investigate the automation of data reliability evaluation. Vulnerability repositories of trusted vendors could be incorporated into a cross-linked local vulnerability database to detect inconsistencies automatically. Incorporating references from additional vulnerability data sources, such as security blogs where vulnerabilities are originally identified, is also beneficial. Advanced natural language processing (Zhu and Dumitras, 2016) and text mining techniques (Murtaza et al., 2016) may be used to automate correlation between numerous vulnerability repositories and correction of conflicting vulnerability information, or to automatically produce vulnerability reports from security forums or news.

### (ii) Further improvement of ML model performance:

To further improve the performance of our ML based vulnerability analysis methods

presented in *Paper III*, *Paper V*, *Paper VII* and Chapter 7, we suggest the following directions:

- The proposed majority voting approach for CVSS ML training ground generation in *Paper III* can be expanded by changing the majority voting tie and incorporating the supervision of experts into the assessment loop. In this approach, security professionals give some startup parameters, and artificial intelligence systems dynamically alter these settings. An alternative is to use the arithmetic mean of scores from various sources to evaluate the dependability of the scores offered by these sources.
- We would extend our model to allow prediction at a more detailed level along the *CWE* hierarchical tree structure. We would also try another method that uses sentimental analysis and sentence similarity assessment for weakness suggestions.
- We plan to test additional ensemble techniques, such as stacking methods, to better differentiate vulnerability instances and to address challenging topics, such as class imbalance problems and computing resource management.

*(iii) Improve further the automation level:*

Automating the query selection process and automating system-level vulnerability assessment are two strategies that can be pursued to raise the level of automation for CI vulnerability analysis. Both of these methods should be examined for potential enhancements. The process of query development described in Chapter 6 requires the assistance of subject matter experts in selecting query tags, a task that can be automated by combining more NLP techniques. Taxonomy and instantiate models for power-grid and manufacturing infrastructures in Chapter 8 extend the application of the provided method and automate the instantiation process. The research conducted in Chapter 8 and Chapter 9 demonstrates the necessity for increased automation in the creation of integrated models. One planned direction is to identify the shared characteristics among a set of generic components, and define “*standard*” components or shared libraries.

*(iv) Assess vulnerability considering environmental and temporal effects:*

A component-level vulnerability index is affected not only by the deployment conditions of the vulnerable component, but also by those of other impacted components. Note that the vulnerable component and the affected component may not be identical, meaning that targeting one component may grant access to another exploit on a different set of components. This is a common trend among APT attack strategies. As reported in *Paper VII*, such vulnerability chains are found and assessed at asset-level or even system-wide level to evaluate the vulnerability implications exposing successful attacks. We define multidimensional functional dependencies that facilitate cascade modeling in Chapter 8. These dependence concepts can also be utilized to calculate the critical level of a component. The suggested taxonomy and instantiated CI models contribute to the establishment of a common framework to give unified knowledge from the views of numerous stakeholders in order to prevent potential risks from penetrating across CI levels and to reduce their impact. On top of the established works, we intend to examine the relationships between vulnerability severity and the attack surface of the system on which the vulnerability assessment is performed.

Multi-level vulnerability assessment would be examined using explainable AI (Liao, Gruen, and Miller, 2020) approaches as part of the intended effort. The objective of this strategy is to generate variable indicators of vulnerability while taking environmental and

temporal effects into account, thereby addressing the challenge of adjusting the necessary information details provided to hierarchical stakeholders in order to achieve an appropriate level of situational awareness. The objective is to condense the information burden relevant to each stakeholder into a form that is less cognitively demanding to understand within the safe remediation decision timeframes of the vulnerability management lifecycle.

From a security evaluation perspective, a critical vulnerability may actually be considered as low patch-prioritization by a system owner possibly due to lower accessibility. Meanwhile, a vulnerability with low impact-severity could lead to catastrophic effect if exploited with chained vulnerabilities. Both scenarios can be systematically analyzed with the extension of our CI architecture and vulnerability retrieval methods.





# APPENDICES

## APPENDIX I: SURVEY ABOUT USER'S PERSPECTIVE ON VULNERABILITY DATABASES AND METRICS

This appendix presents a survey submitted to CI cybersecurity stakeholders who have basic knowledge of CI cybersecurity. The surveys were distributed online in 2019. The ones attached in this survey are a partial copy of the original ones, as some questions included in the original ones are not related to this thesis. Some important survey results are presented in the end of this appendix.

### **Industrial Cyber-Security Survey**

Thank you for taking part in this survey and for contributing to our research!

*Aim:* The aim of this survey is to gather data about your perception of Industrial Cyber-Security Protection. That means, your perception of the threats and vulnerability that shall be prioritized in industry risk management, as well as of using open-source security databases, vulnerability scoring mechanisms and artificial intelligence to support industrial vulnerability analysis and decision making.

*Anonymity:* The information obtained from this questionnaire will be employed for research purposes and the anonymity of the participants is ensured. No individual information will be disclosed. Please do not write your name in the survey.

*Duration:* The questionnaire will take about 10-15 minutes to complete.

*Contact:* Should you wish us to contact you or any member of your organization for any further discussion about the above, you may contact:

Yuning Jiang (PhD candidate)  
Distributed Real-Time System  
School of Informatics, University of Skövde  
Email: yuning.jiang@his.se

## Participant Information

**1. What are the following descriptions best describing your field? (Choose one or more options).**

- Industrial control system security.
- Network security.
- IT services.
- Telecommunication operations/equipments.
- Cybersecurity research.
- Manufacturing industry.
- Software development.
- Other: \_\_\_\_\_

**2. What are the following descriptions best describing your role? (Choose one or more options).**

- Technician/Analyst.
- Production staff.
- System operator.
- Manager.
- Consultant.
- Teacher.
- Student.
- Other: \_\_\_\_\_

### Using Public Cybersecurity Data Sources and Metrics for CI Cybersecurity

**3. To what extent open-source security databases, e.g. vulnerability repository CVE and exploit database ExploitDB are used in critical infrastructures to provide vulnerability alerts? (Select one option below.)**

None       Low       Medium       High

**4. How often do you use open-source security databases, e.g., vulnerability repository CVE and exploit database ExploitDB? (Select one frequency below.)**

Never     Once a year     Once a month     Once a week     Everyday

- 5. Please rank the following critical infrastructure vulnerability management strategies in the order of importance**
- Structure-level vulnerability analysis to improve resilience.
  - Product-level vulnerability analysis to prioritise patching.
  - Provide frequent penetration testing.
  - Cyber vulnerability intelligence to support vulnerability prediction.
  - Other: \_\_\_\_\_
- 6. What are the purposes for you to use the open-source security databases, e.g., vulnerability database CVE and exploit database ExploitDB? (Select one or more options below. If the motivation to use open-source security databases is not listed, please write it in Other.)**
- To know the general security status.
  - To know the latest security status.
  - To compare and choose from products from different vendors.
  - To know the security status of my products
  - Other: \_\_\_\_\_
- 7. To what extent are the industrial vulnerability scoring mechanisms, e.g., common vulnerability scoring system (CVSS), used in critical infrastructures to support vulnerability analysis? (Select one option below.)**
- None                    Low                    Medium                    High
- 8. How often do you use the industrial vulnerability scoring mechanisms, e.g. common vulnerability scoring system (CVSS)? (Select one frequency below)**
- Never    Once a year    Once a month    Once a week    Everyday
- 9. What are the purposes for you to use the industrial vulnerability scoring mechanisms, e.g., common vulnerability scoring system (CVSS)?(Select one or more options below. If the motivation to use open-source security databases is not listed, please write it in Other.)**
- To evaluate a security issue.
  - To compare and choose from products from different vendors.
  - To know the security status of my products
  - Other: \_\_\_\_\_

Figure 10.2 presents results on correlations between working field and usage of CVE.

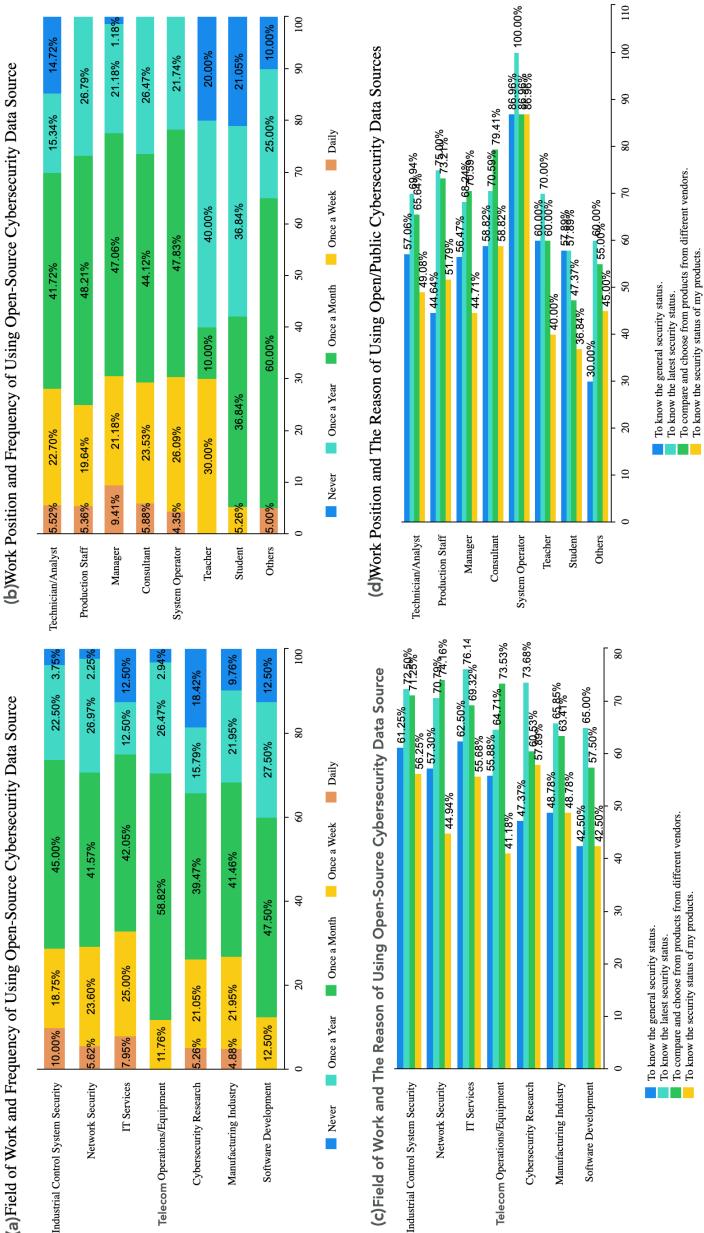


Figure 10.2: Correlation between working field and usage of CVE

Figure 10.3 presents results on correlations between working field and usage of CVSS.

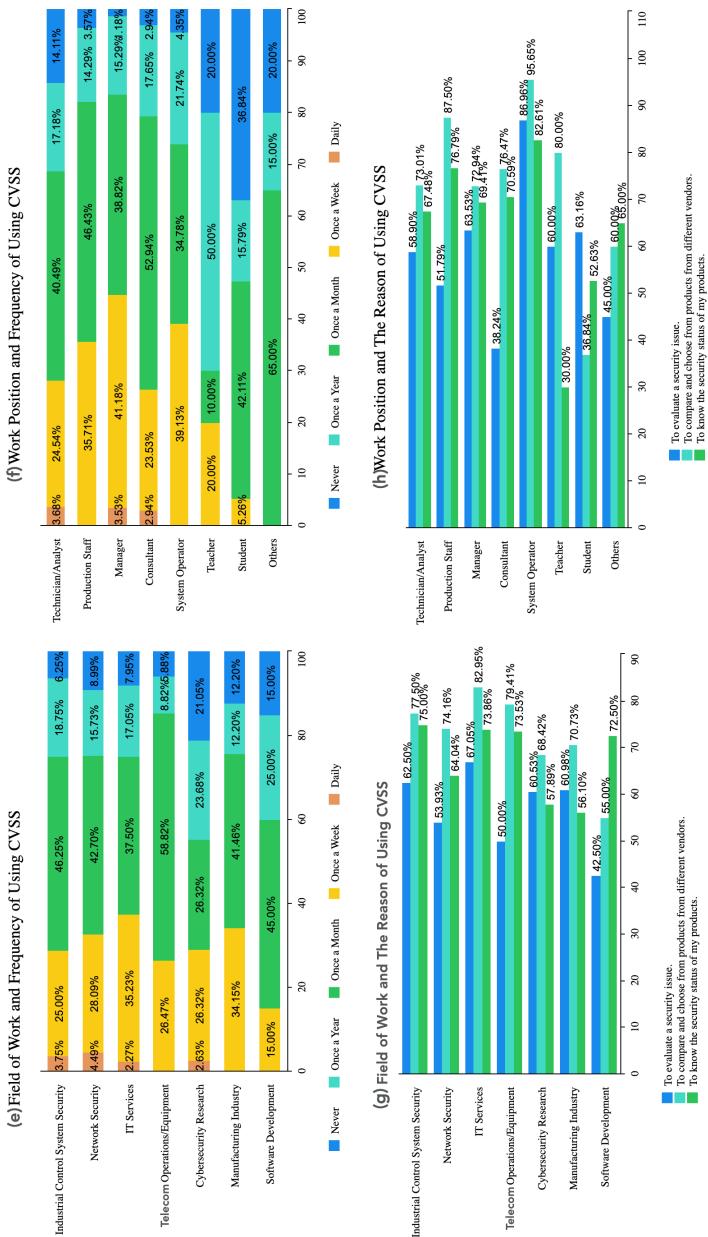


Figure 10.3: Correlation between working field and usage of CVSS

## APPENDIX II: INTERVIEW QUESTIONNAIRE FOR *STUDY III*

This appendix presents the questions that led the discussions during the interviews performed in *Study III*. Summaries of the interview results are presented in Chapter 5. Although a list of questions was prepared before the interview, it was conducted in a semi-structure manner, opening opportunities to both wider and deeper discussions related vulnerability-driven cybersecurity practices and challenges in complicated critical infrastructures.

Several interviews were conducted at different stages of *Study III*. More specifically, interview performed at an earlier stage of this case study aims to understand the vulnerability assessment and management practices in the involved organization. The interview carried out in the end of this case study validates the usefulness and applicability of public cybersecurity databases based on results of *Study III*.

### INTERVIEW QUESTIONNAIRE I

**Objective:** to gain knowledge about the organization and specifically, about the vulnerability-assessment process in this organization.

#### ***Understanding the organization:***

- What are the current challenges of the vulnerability assessment in your organization?
- What are your requirements for vulnerability assessment?
- How is vulnerability assessment related to security protection in your organization?
- Do you think that you have a comprehensive and up-to-date overview of the vulnerability situation of your IT/OT systems?
- How often do you conduct vulnerability assessment of your system?
- How often do you receive vulnerability alerts, considering the events triggered alarms of your system?
- How long does it take for you between receiving vulnerability alerts and carrying patching actions?
- How is your vulnerability assessment conducted? Manually or Automatically?
- Is there any part of your vulnerability-assessment process that needs further automation?

#### ***Public Vulnerability Database and Severity Metrics:***

- To what extent/percentage open-source security databases, e.g vulnerability repository CVE and exploit database ExploitDB are used in your organization?
- Which open-source security databases are you using?
- What are the purposes for you to use the open-source security databases?

- To what extent are the industrial vulnerability scoring mechanisms, e.g. common vulnerability scoring system (CVSS), used in your organization?
- What are the purposes for you to use CVSS or other scoring systems?

## INTERVIEW QUESTIONNAIRE II

**Objective:** to get feedback on the results of *Study III* and specifically, comments upon using public accessible vulnerability data sources.

### ***Identified Vulnerabilities:***

- Is the generated vulnerability report a useful aid for cybersecurity decision making of the investigated system?
- Do the case study results (and the presentation of these findings) make you aware of weaknesses that you did not know before?
- Do you agree with the top 3 weaknesses identified in this case study? Which rank is most wrong from your point of view? (Note that the top 3 weaknesses are the ones with the highest occurrences in the investigated system.)
- When you make cybersecurity decisions, would you rely more on the vulnerability-assessment results using *NVD* or vendor entries? Or both?

## APPENDIX III: EXAMPLE OF VULNERABILITY DATABASE OUTPUT IN JSON

This appendix presents a vulnerability instance CVE-2021-36745 exported from our proposed vulnerability database in JSON format. Its relevant information is composed of eight nested documents, namely *Vulnerability* (basic metadata), *Tracking*, *Threat*, *Weakness*, *Attack*, *Severity* and *Affected Product* (marked in red color).

```
1  {
2      "vulnerability_metadata": {
3          "id": [ {
4              "id_data" : "CVE-2021-36745",
5              "source" : "nvd"
6          }, {
7              "id_data" : "ZDI-21-1115",
8              "source" : "zero day initialitive"
9          }
10     ],
11     "assigner": "security@trendmicro.com",
12     "report": [ {
13         "summary" : "A vulnerability in Trend Micro ServerProtect for Storage 6.0, ✓
14             ServerProtect for EMC Celerra 5.8, ServerProtect for Network Appliance ✓
15             Filers 5.8, and ServerProtect for Microsoft Windows / Novell Netware 5.8 ✓
16             could allow a remote attacker to bypass authentication on affected ✓
17             installations.",
18             "source" : "nvd"
19     }, {
20         "summary" : "This vulnerability allows remote attackers to bypass ✓
21             authentication on affected installations of Trend Micro ServerProtect. ✓
22             Authentication is not required to exploit this vulnerability. The ✓
23             specific flaw exists within the ServerProtect console. The issue results ✓
24             from the lack of proper validation prior to authentication. An attacker ✓
25             can leverage this vulnerability to bypass authentication on the system.",
26             "source" : "zero day initialitive"
27     }
28 ],
29     "references": {
30         "reference_data" : [ {
31             "url" : "https://success.trendmicro.com/solution/000289038",
32             "tags" : [ "Patch", "Vendor Advisory" ]
33         }, {
34             "url" : "https://www.zerodayinitiative.com/advisories/ZDI-21-1115/",
35             "tags" : [ "Third Party Advisory", "VDB Entry" ]
36         }, {
37             "url" : "https://success.trendmicro.com/jp/solution/000289030",
38             "tags" : [ "Patch", "Vendor Advisory" ]
39         }
40     ]
41 },
42     "tracking": {
43         "disclosure": [ {
44             "disclose" : "2021-04-14",
45             "source" : "trend micro"
46         } ],
47         "publication": [ {
48             "published" : "2021-09-29",
49             "source" : "nvd"
50         }, {
51             "published" : "2021-09-24",
52             "source" : "trend micro"
53         }
54     ],
55         "modification": [ {
56             "modified" : "2021-10-02",
57             "source" : "nvd"
58         } ],
59         "patch": [ {
60             "patch_released" : "2021-09-24",
61             "source" : "trend micro"
62         }
63     ]
64 }
```

```

53     } ]
54 },
55 "threat": {
56     "cvedetails": [
57         "cvedetails_data" : "Bypass",
58         "source" : "cvedetails"
59     ]
60 },
61 "weakness": [
62     "cwe": [
63         "cwe_id" : "CWE-287",
64         "cwe_name": "Improper Authentication",
65         "cwe_description": "When an actor claims to have a given identity, the ↴
66             software does not prove or insufficiently proves that the claim is ↴
67                 correct.",
68         "cwe_exploit_likelihood": "High",
69         "source" : "nvd"
70     ]
71 },
72 "attack": [
73     "capec_data": [
74         "capec_id" : "CAPEC-114",
75         "capec_name": "Authentication Abuse",
76         "capec_typical_severity": "Medium",
77         "capec_prerequisites": "An authentication mechanism or subsystem ↴
78             implementing some form of authentication such as passwords, digest ↴
79                 authentication, security certificates, etc. which is flawed in some way." ↴
80         ,
81         "capec_resource_required": "A client application, command-line access to a ↴
82             binary, or scripting language capable of interacting with the ↴
83                 authentication mechanism.",
84         "source" : "nvd"
85     ],
86     "capec_id" : "CAPEC-115",
87     "capec_name": "Authentication Bypass",
88     "capec_typical_severity": "Medium",
89     "capec_prerequisites": "An authentication mechanism or subsystem ↴
90             implementing some form of authentication such as passwords, digest ↴
91                 authentication, security certificates, etc.",
92     "capec_resource_required": "A client application, such as a web browser, ↴
93             or a scripting language capable of interacting with the target.",
94     "source" : "nvd"
95 },
96     "capec_id" : "CAPEC-151",
97     "capec_name": "Identity Spoofing",
98     "capec_typical_severity": "Medium",
99     "capec_prerequisites": "The identity associated with the message or ↴
100             resource must be removable or modifiable in an undetectable way.",
101     "capec_resource_required": "None: No specialized resources are required ↴
102             to execute this type of attack.",
103     "source" : "nvd"
104 },
105     "capec_id" : "CAPEC-194",
106     "capec_name": "Fake the Source of Data",
107     "capec_typical_severity": "Medium",
108     "capec_prerequisites": "This attack is only applicable when a vulnerable ↴
109             entity associates data or services with an identity. Without such an ↴
110                 association, there would be no reason to fake the source.",
111     "capec_resource_required": "Resources required vary depending on the ↴
112             nature of the attack. Possible tools needed by an attacker could ↴
113                 include tools to create custom network packets, specific client ↴
114                     software, and tools to capture network traffic. Many variants of this ↴
115                         attack require no attacker resources, however.",
116     "source" : "nvd"
117 },
118     "capec_id" : "CAPEC-22",
119     "capec_name": "Exploiting Trust in Client",
120     "capec_typical_severity": "High",
121     "capec_prerequisites": "Server software must rely on client side ↴
122             formatted and validated values, and not reinforce these checks on the ↴
123                 server side.",
124     "capec_resource_required": "Ability to communicate synchronously or ↴
125                     asynchronously with server.",

```

```

105     "source" : "nvd"
106   },
107   {
108     "capec_id" : "CAPEC-57",
109     "capec_name": "Utilizing REST's Trust in the System Resource to Obtain ↵
110       Sensitive Data",
111     "capec_typical_severity": "Very High",
112     "capec_prerequisites": "Opportunity to intercept must exist beyond the ↵
113       point where SSL is terminated. The attacker must be able to insert a ↵
114       listener actively (proxying the communication) or passively (sniffing ↵
115       the communication) in the client-server communication path.",
116     "capec_resource_required": "",
117     "source" : "nvd"
118   },
119   {
120     "capec_id" : "CAPEC-593",
121     "capec_name": "Session Hijacking",
122     "capec_typical_severity": "Very High",
123     "capec_prerequisites": "An application that leverages sessions to perform ↵
124       authentication.",
125     "capec_resource_required": "The adversary must have the ability to ↵
126       communicate with the application over the network.",
127     "source" : "nvd"
128   },
129   {
130     "capec_id" : "CAPEC-633",
131     "capec_name": "Token Impersonation",
132     "capec_typical_severity": "Medium",
133     "capec_prerequisites": "This pattern of attack is only applicable when a ↵
134       downstream user leverages tokens to verify identity, and then takes ↵
135       action based on that identity.",
136     "capec_resource_required": "",
137     "source" : "nvd"
138   },
139   {
140     "capec_id" : "CAPEC-650",
141     "capec_name": "Upload a Web Shell to a Web Server",
142     "capec_typical_severity": "High",
143     "capec_prerequisites": "The web server is susceptible to one of the ↵
144       various web application exploits that allows for uploading a shell file ↵
145       .",
146     "capec_resource_required": "",
147     "source" : "nvd"
148   },
149   {
150     "capec_id" : "CAPEC-94",
151     "capec_name": "Adversary in the Middle (AiTM)",
152     "capec_typical_severity": "Very High",
153     "capec_prerequisites": "There are two components communicating with each ↵
154       other. An attacker is able to identify the nature and mechanism of ↵
155       communication between the two target components. An attacker can ↵
156       eavesdrop on the communication between the target components. Strong ↵
157       mutual authentication is not used between the two target components ↵
158       yielding opportunity for attacker interposition. The communication ↵
159       occurs in clear (not encrypted) or with insufficient and spoofable ↵
160       encryption.",
161     "capec_resource_required": "",
162     "source" : "nvd"
163   }
164 ],
165 "attck_data": [
166   {
167     "attck_id": "T1548",
168     "attck_name": "Abuse Elevation Control Mechanism",
169     "attck_tactic": ["Privilege Escalation", "Defense Evasion"],
170     "attck_mitigation": [
171       {
172         "attck_mitigation_id": "M1047",
173         "attck_mitigation_name": "Audit"
174       },
175       {
176         "attck_mitigation_id": "M1038",
177         "attck_mitigation_name": "Execution Prevention"
178       },
179       {
180         "attck_mitigation_id": "M1028",
181         "attck_mitigation_name": "Operating System Configuration"
182       },
183       {
184         "attck_mitigation_id": "M1026",
185         "attck_mitigation_name": "Privileged Account Management"
186       },
187       {
188         "attck_mitigation_id": "M1022",
189       }
190     ]
191   }
192 ]

```

```

161         "attck_mitigation_name": "Restrict File and Directory Permissions"
162     },{
163         "attck_mitigation_id": "M1052",
164         "attck_mitigation_name": "User Account Control"
165     }],
166     "source" : "nvd"
167 },{
168     "attck_id": "T1550.001",
169     "attck_name": "Use Alternate Authentication Material: Application Access Token",
170     "attck_tactic": ["Lateral Movement", "Defense Evasion"],
171     "attck_mitigation": [
172         {
173             "attck_mitigation_id": "M1047",
174             "attck_mitigation_name": "Audit"
175         },
176         {
177             "attck_mitigation_id": "M1041",
178             "attck_mitigation_name": "Encrypt Sensitive Information"
179         },
180         {
181             "attck_mitigation_id": "M1021",
182             "attck_mitigation_name": "Restrict Web-Based Content"
183         },
184         {
185             "attck_id": "T1563",
186             "attck_name": "Remote Service Session Hijacking",
187             "attck_tactic": ["Lateral Movement"],
188             "attck_mitigation": [
189                 {
190                     "attck_mitigation_id": "M1042",
191                     "attck_mitigation_name": "Disable or Remove Feature or Program"
192                 },
193                 {
194                     "attck_mitigation_id": "M1030",
195                     "attck_mitigation_name": "Network Segmentation"
196                 },
197                 {
198                     "attck_mitigation_id": "M1026",
199                     "attck_mitigation_name": "Privileged Account Management"
200                 },
201                 {
202                     "attck_mitigation_id": "M1018",
203                     "attck_mitigation_name": "User Account Management"
204                 },
205                 {
206                     "attck_mitigation_id": "M1026",
207                     "attck_mitigation_name": "Privileged Account Management"
208                 },
209                 {
210                     "attck_mitigation_id": "M1018",
211                     "attck_mitigation_name": "User Account Management"
212                 },
213                 {
214                     "attck_id": "T1134",
215                     "attck_name": "Access Token Manipulation",
216                     "attck_tactic": ["Privilege Escalation", "Defense Evasion"],
217                     "attck_mitigation": [
218                         {
219                             "attck_mitigation_id": "M1026",
220                             "attck_mitigation_name": "Privileged Account Management"
221                         },
222                         {
223                             "attck_mitigation_id": "M1018",
224                             "attck_mitigation_name": "User Account Management"
225                         }
226                     ],
227                     "source" : "nvd"
228                 },
229                 {
230                     "attck_id": "T1505.003",
231                     "attck_name": "Server Software Component: Web Shell",
232                     "attck_tactic": ["Persistence"],
233                     "attck_mitigation": [
234                         {
235                             "attck_mitigation_id": "M1042",
236                             "attck_mitigation_name": "Disable or Remove Feature or Program"
237                         },
238                         {
239                             "attck_mitigation_id": "M1018",
240                             "attck_mitigation_name": "User Account Management"
241                         }
242                     ],
243                     "source" : "nvd"
244                 },
245                 {
246                     "attck_id": "T1557",
247                     "attck_name": "Adversary-in-the-Middle",
248                     "attck_tactic": ["Collection", "Credential Access"],
249                     "attck_mitigation": [
250                         {
251                             "attck_mitigation_id": "M1042",
252                             "attck_mitigation_name": "Disable or Remove Feature or Program"
253                         }
254                     ]
255                 }
256             ]
257         ]
258     ]
259 }

```

```

232         "attck_mitigation_id": "M1041",
233         "attck_mitigation_name": "Encrypt Sensitive Information"
234     },
235     "attck_mitigation_id": "M1037",
236     "attck_mitigation_name": "Filter Network Traffic"
237 },
238     "attck_mitigation_id": "M1035",
239     "attck_mitigation_name": "Limit Access to Resource Over Network"
240 },
241     "attck_mitigation_id": "M1031",
242     "attck_mitigation_name": "Network Intrusion Prevention"
243 },
244     "attck_mitigation_id": "M1030",
245     "attck_mitigation_name": "Network Segmentation"
246 },
247     "attck_mitigation_id": "M1017",
248     "attck_mitigation_name": "User Training"
249 ],
250     "source" : "nvd"
251   ]
252 },
253   "remediation": [
254     "remediation_data": [
255       {
256         "update_version" : [
257           "ServerProtect for EMC Celerra (SPEMC) Update to 5.8CP1577",
258           "ServerProtect for Storage (SPFS) Update to 6.0 CP1284",
259           "ServerProtect for Network Appliance Filers (SPNAF) Update to 5.8CP1299",
260           "",
261           "ServerProtect for Microsoft Windows / Novell Netware (SPNT) Update to 5.8CP1575"
262         ],
263         "source" : "trend micro"
264       }
265     ],
266     "severity": [
267       "cvss_v3": [
268         {
269           "base_score": 9.8,
270           "base_vector": "AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H",
271           "attackcomplexity": "Low",
272           "attackvector": "Network",
273           "privilegesrequired": "None",
274           "userinteraction": "None",
275           "scope": "Unchanged",
276           "confidentiality": "High",
277           "integrity": "High",
278           "availability": "High",
279           "source" : "nvd"
280         },
281         {
282           "base_score": 9.8,
283           "base_vector": "AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H",
284           "attackcomplexity": "Low",
285           "attackvector": "Network",
286           "privilegesrequired": "None",
287           "scope": "Unchanged",
288           "userinteraction": "None",
289           "availability": "High",
290           "confidentiality": "High",
291           "integrity": "High",
292           "source" : "trend micro"
293         }
294       ],
295       "cvss_v2": [
296         {
297           "base_score": 10.0,
298           "base_vector": "AV:N/AC:L/Au:N/C:C/I:C/A:C",
299           "accessVector" : "Network",
300           "accessComplexity" : "Low",
301           "authentication" : "None",
302           "confidentialityImpact" : "Complete",
303           "integrityImpact" : "Complete",
304           "availabilityImpact" : "Complete",
305           "source" : "nvd"
306         }
307       ]
308     ]
309   ]
310 }
```

```

302 },
303   "affected_products": {
304     "cve_data": {
305       "affected" : [
306         "Trend Micro - ServerProtect for Storage - 6.0",
307         "Trend Micro - ServerProtect for EMC Celerra - 5.8",
308         "Trend Micro - ServerProtect for Network Appliance Filers - 5.8",
309         "Trend Micro - ServerProtect for Microsoft Windows / Novell Netware - 5.8"
310       ],
311       "source" : "cve"
312     },
313     "cpe_data": {
314       "nodes" : [ {
315         "operator" : "OR",
316         "children" : [ ],
317         "cpe_match" : [ {
318           "vulnerable" : true,
319           "cpe23Uri" : "cpe:2.3:a:trendmicro:serverprotect:5.8:*:*:*:*:emc:*:*",
320           "cpe_name" : [ ]
321         }, {
322           "vulnerable" : true,
323           "cpe23Uri" : "cpe:2.3:a:trendmicro:serverprotect:5.8:*:*:*:*:netapp:*:*",
324           "cpe_name" : [ ]
325         }, {
326           "vulnerable" : true,
327           "cpe23Uri" : "cpe:2.3:a:trendmicro:serverprotect:5.8:*:*:*:*:netware:*:*",
328           "cpe_name" : [ ]
329         }, {
330           "vulnerable" : true,
331           "cpe23Uri" : "cpe:2.3:a:trendmicro:serverprotect:5.8:*:*:*:*:windows:*:*",
332           "cpe_name" : [ ]
333         }, {
334           "vulnerable" : true,
335           "cpe23Uri" : "cpe:2.3:a:trendmicro:serverprotect:6.0:*:*:*:*:storage:*:*",
336           "cpe_name" : [ ]
337         } ]
338       },
339       "source" : "nvd"
340     },
341     "vendor_data": {
342       "affected" : [
343         "ServerProtect for EMC Celerra (SPEMC) 5.8",
344         "ServerProtect for Storage (SPFS) 6.0",
345         "ServerProtect for Network Appliance Filers (SPNAF) 5.8",
346         "ServerProtect for Microsoft Windows / Novell Netware (SPNT) 5.8"
347       ],
348       "source" : "trend micro"
349     }
350   }
351 }
```

#### APPENDIX IV: INSTANTIATED MODELS FOR STUDY X

Figure 10.4 presents the cyber and control layers of *Model I* used in *Study X*.

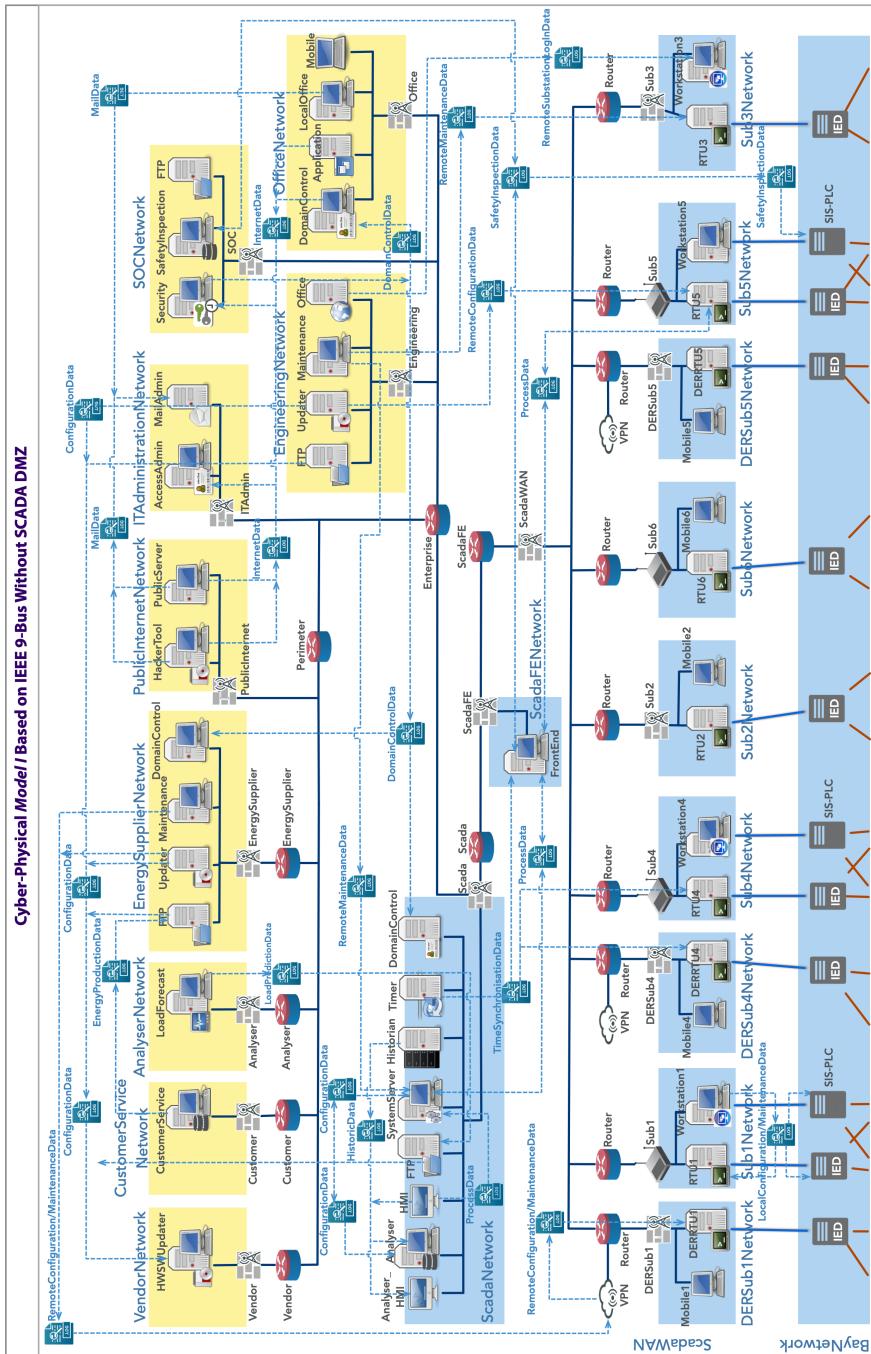


Figure 10.4: Cyber and control layers of *Model I* used in *Study X*

Figure 10.5 presents the physical layer of *Model I* used in *Study X*.

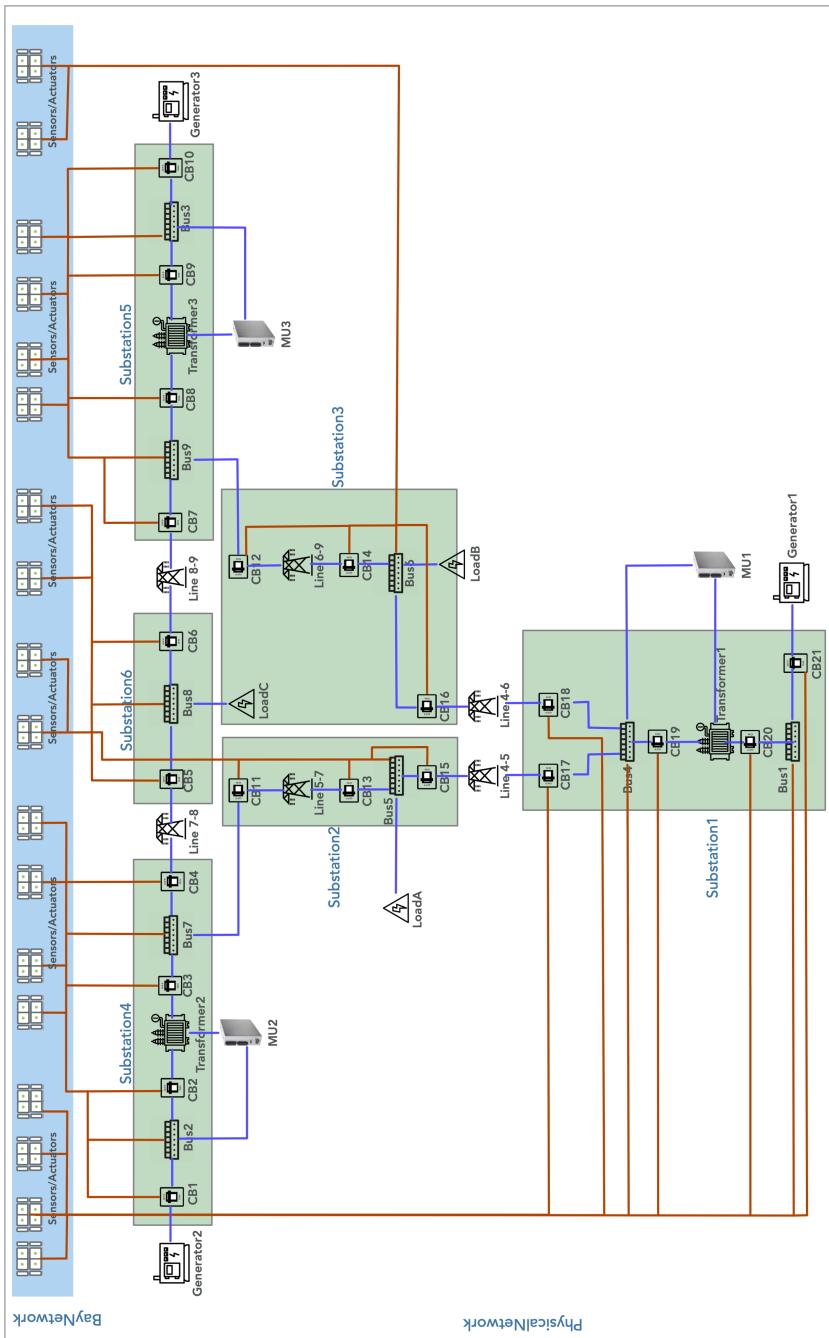


Figure 10.5: Physical layer of *Model I* used in *Study X*

Figure 10.6 presents the cyber and control layers of *Model II* used in *Study X*.

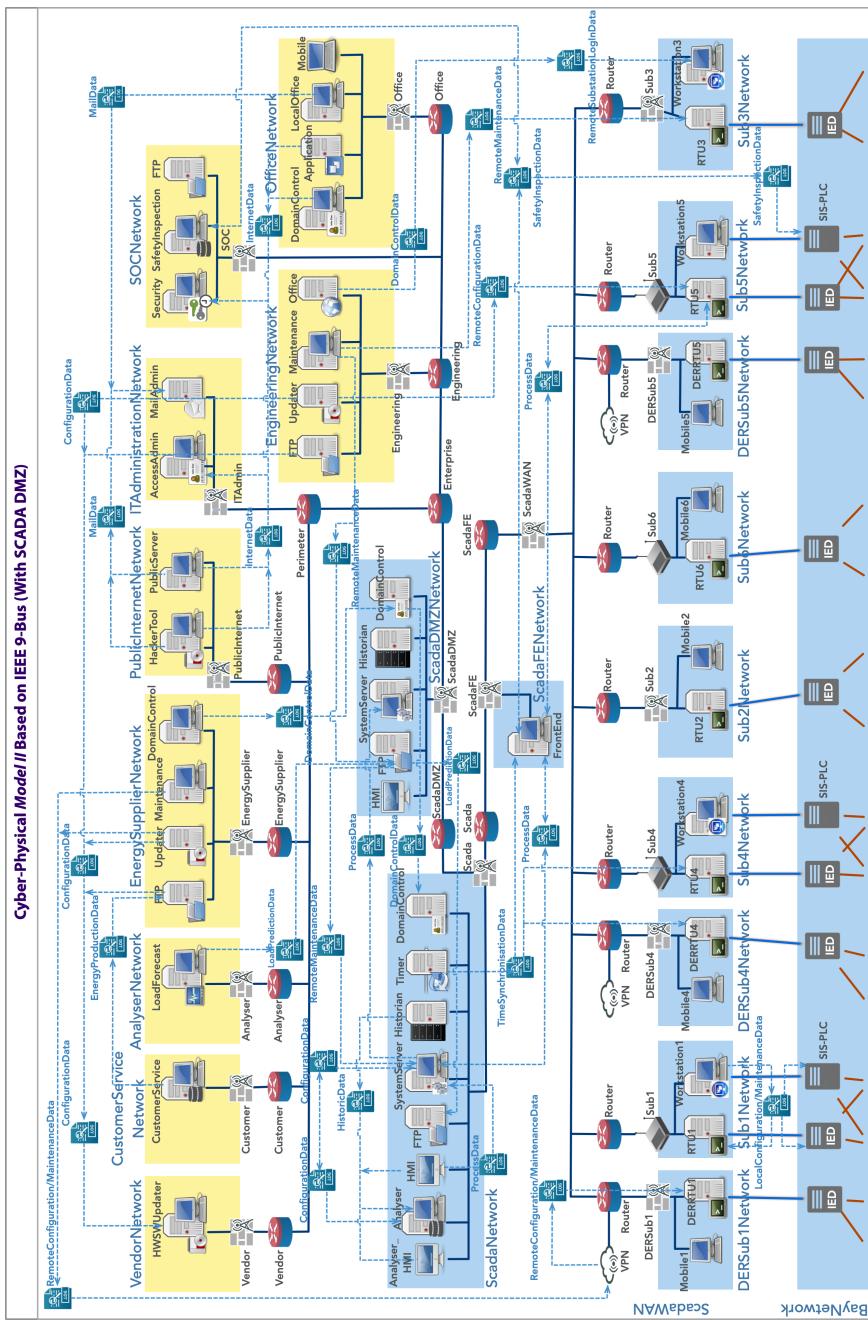


Figure 10.6: Cyber and control layers of *Model II* used in *Study X*

Figure 10.7 presents the physical layer of *Model II* used in *Study X*.

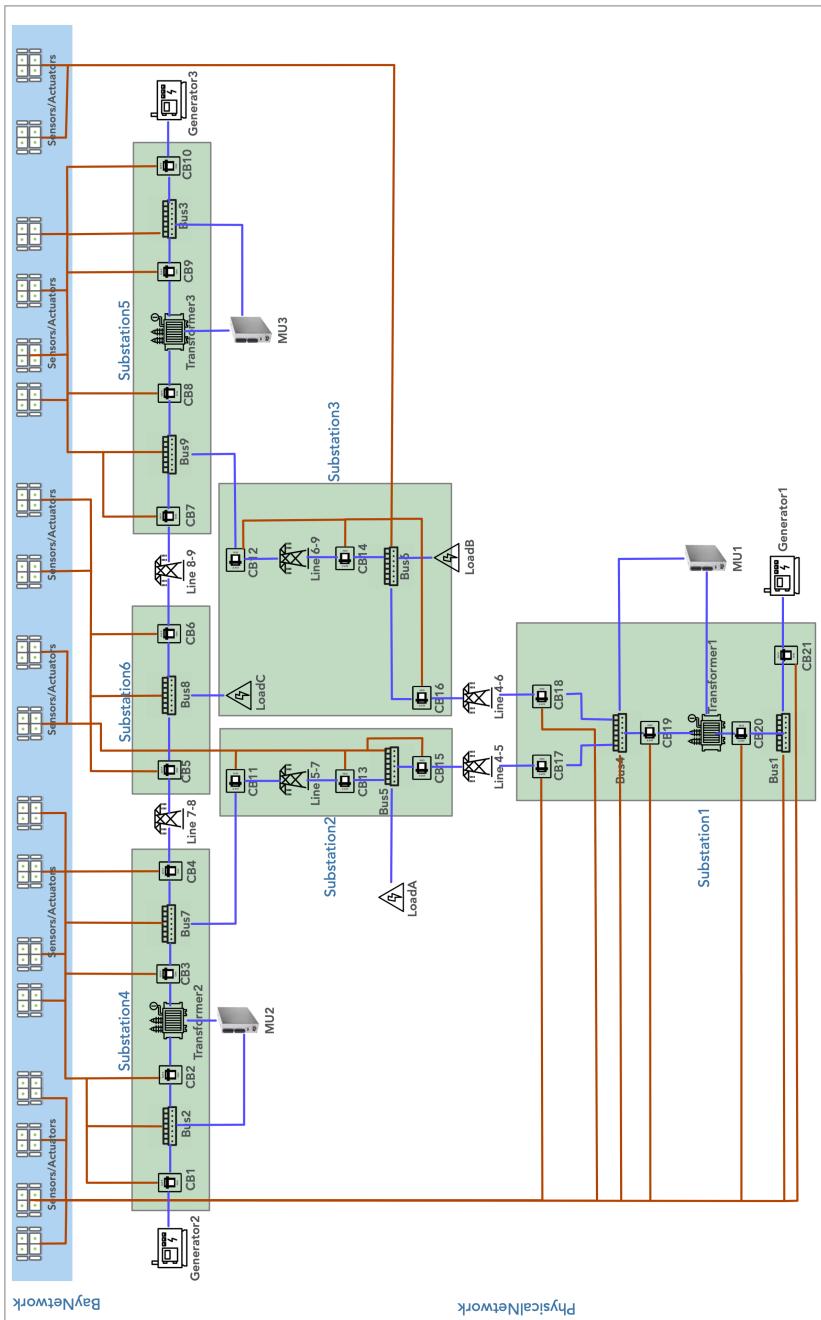


Figure 10.7: Physical layer of *Model II* used in *Study X*







## REFERENCES

- 2021 Top Routinely Exploited Vulnerabilities (2022). <https://www.cisa.gov/uscert/ncas/alerts/aa22-117a>. Accessed: 2022-04-23.
- ABB Energy Manager (2022). <https://new.abb.com/industrial-software/sustainability/energy-manager/industrial-energy-load-planning-forecasting-scheduling>. Accessed: 2022-04-23.
- Abbott, Robert P, Chin, Janet S, Donnelley, James E, Konigsford, William L, Tokubo, S, and Webb, Douglas A (1976). *Security Analysis and Enhancements of Computer Operating Systems*. Tech. rep. National Bureau of Standards Washington DC Inst For Computer Sciences and Technology.
- Abubakar, I, Khalid, SN, Mustafa, MW, Shareef, Hussain, and Mustapha, M (2017). “Application of load monitoring in appliances energy management—A review.” In: *Renewable and Sustainable Energy Reviews* 67, pp. 235–245.
- Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) (2022). <https://attack.mitre.org/>. Accessed: 2022-04-30.
- Aghaei, Ehsan, Shadid, Waseem, and Al-Shaer, Ehab (2020). “ThreatZoom: Hierarchical Neural Network for CVEs to CWEs Classification.” In: *International Conference on Security and Privacy in Communication Systems*. Springer, pp. 23–41.
- Aghaei, Ehsan and Al-Shaer, Ehab (2019). “Threatzoom: Neural Network for Automated Vulnerability Mitigation.” In: *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security*, pp. 1–3.
- Agyepong, Enoch, Cherdantseva, Yulia, Reinecke, Philipp, and Burnap, Pete (2019). “Challenges and Performance Metrics for Security Operations Centre Analysts: A Systematic Review.” In: *Journal of Cyber Security Technology*, pp. 1–28.
- Avatoglou, Georgios, Anastasiadis, Mike, Spanos, Georgios, Voulgaridis, Antonis, Votis, Konstantinos, and Tzovaras, Dimitrios (2021). “A Tree-Based Machine Learning Methodology to Automatically Classify Software Vulnerabilities.” In: *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, pp. 312–317.
- Akbarzadeh, Aida and Katsikas, Sokratis (2021). “Identifying and Analyzing Dependencies in and among Complex Cyber Physical Systems.” In: *Sensors* 21.5, p. 1685.
- Alberts, Christopher, Dorofee, Audrey, Stevens, James, and Woody, Carol (2003). *Introduction to the OCTAVE Approach*. Tech. rep. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

- Allodi, Luca and Massacci, Fabio (2014). "Comparing Vulnerability Severity and Exploits Using Case-Control Studies." In: *ACM Transactions on Information and System Security (TISSEC)* 17.1, pp. 1–20.
- Allodi, Luca and Massacci, Fabio (2017). "Security Events and Vulnerability Data for Cybersecurity Risk Estimation." In: *Risk Analysis* 37.8, pp. 1606–1627.
- Almukaynizi, Mohammed, Nunes, Eric, Dharaiya, Krishna, Senguttuvan, Manoj, Shakarian, Jana, and Shakarian, Paulo (2017). "Proactive Identification of Exploits in the Wild Through Vulnerability MentionsOnline." In: *2017 International Conference on Cyber Conflict (CyCon US)*. IEEE, pp. 82–88.
- Alqahtani, Sultan S, Eghan, Ellis E, and Rilling, Juergen (2016a). "SV-AF - A Security Vulnerability Analysis Framework." In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, pp. 219–229.
- Alqahtani, Sultan S, Eghan, Ellis E, and Rilling, Juergen (2016b). "Tracing Known Security Vulnerabilities in Software Repositories—A Semantic WebEnabled Modelling Approach." In: *Science of Computer Programming* 121, pp. 153–175.
- Aly, Mohamed (2005). "Survey on Multiclass Classification Methods." In: *Neural Netw* 19, pp. 1–9.
- Ammann, Paul, Wijesekera, Duminda, and Kaushik, Saket (2002). "Scalable, Graph-Based Network Vulnerability Analysis." In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 217–224.
- Andersen, Hanne and Hepburn, Brian (2016). "Scientific Method." In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2016. Metaphysics Research Lab, Stanford University.
- Andrade, Roberto Omar, Fuertes, Walter, Cadena, Susana, Cadena, Alyssa, Tello-Oquendo, Luis, Córdova, Daniela, Garcés, Iván Ortiz, and Cazares, Mariña Fernanda (2019). "Information Security Management in University Campus Using Cognitive Security." In: *International Journal of Computer Science and Security (IJCSS)* 13.4, p. 124.
- Ani, Uchenna P Daniel, He, Hongmei, and Tiwari, Ashutosh (2017). "Review of Cybersecurity Issues in Industrial Critical Infrastructure: Manufacturing in Perspective." In: *Journal of Cyber Security Technology* 1.1, pp. 32–74.
- Antunes, Nuno and Vieira, Marco (2014). "Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples." In: *IEEE Transactions on Services Computing* 8.2, pp. 269–283.
- Anwar, Afsah, Abusnaina, Ahmed, Chen, Songqing, Li, Frank, and Mohaisen, David (2020). "Cleaning the NVD: Comprehensive Quality Assessment, Improvements, and Analyses." In: *arXiv preprint arXiv:2006.15074*.
- Arbaugh, William A, Fithen, William L, and McHugh, John (2000). "Windows of Vulnerability: A Case Study Analysis." In: *Computer* 33.12, pp. 52–59.

- Arora, Ashish, Krishnan, Ramayya, Telang, Rahul, and Yang, Yubao (2006). “An Empirical Analysis of Software Vendors’ Patching Behavior: Impact of Vulnerability Disclosure.” In: *ICIS 2006 Proceedings*, p. 22.
- Arp, Daniel, Spreitzenbarth, Michael, Hubner, Malte, Gascon, Hugo, Rieck, Konrad, and Siemens, CERT (2014). “Drebin: Effective and Explainable Detection of Android Malware in Your Pocket.” In: *Ndss*. Vol. 14, pp. 23–26.
- Asghar, Muhammad Rizwan, Hu, Qinwen, and Zeadally, Sherali (2019). “Cybersecurity in Industrial Control Systems: Issues, Technologies, and Challenges.” In: *Computer Networks* 165, p. 106946.
- Ashibani, Yosef and Mahmoud, Qusay H (2017). “Cyber Physical Systems Security: Analysis, Challenges and Solutions.” In: *Computers & Security* 68, pp. 81–97.
- Austin, Andrew, Holmgreen, Casper, and Williams, Laurie (2013). “A Comparison of the Efficiency and Effectiveness of Vulnerability Discovery Techniques.” In: *Information and Software Technology* 55.7, pp. 1279–1288.
- Barnaghi, Payam, Wang, Wei, Henson, Cory, and Taylor, Kerry (2012). “Semantics for the Internet of Things: Early Progress and back to the Future.” In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 8.1, pp. 1–21.
- Beach-Westmoreland, Nate, Styczynski, Jake, and Stables, Scott (Nov. 2016). *When The Lights Went Out*. Tech. rep. Booz Allen Hamilton. url: <https://www.boozallen.com/content/dam/boozallen/documents/2016/09/ukraine-report-when-the-lights-went-out.pdf>.
- Beautiful Soup* (2022). <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>. Accessed: 2022-04-23.
- Benzel, Terry (2011). “The Science of Cyber Security Experimentation: the DETER Project.” In: *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 137–148.
- Berger, Paul D and Maurer, Robert E (2002). “Experimental Design.” In: *CA (USA): Wadsworth Group Belmont*.
- Bernstein, Philip A and Haas, Laura M (2008). “Information integration in the enterprise.” In: *Communications of the ACM* 51.9, pp. 72–79.
- Bertoglio, Daniel Dalalana and Zorzo, Avelino Francisco (2017). “Overview and Open Issues on Penetration Test.” In: *Journal of the Brazilian Computer Society* 23.1, p. 2.
- Bevan, Nigel, Carter, James, and Harker, Susan (2015). “ISO 9241-11 revised: What have we learnt about usability since 1998?” In: *International conference on human-computer interaction*. Springer, pp. 143–151.
- Bevan, Nigel, Carter, Jim, Earthy, Jonathan, Geis, Thomas, and Harker, Susan (2016). “New ISO standards for usability, usability reports and usability measures.” In: *International conference on human-computer interaction*. Springer, pp. 268–278.

- Bhamare, Deval, Zolanvari, Maede, Erbad, Aiman, Jain, Raj, Khan, Khaled, and Me-skin, Nader (2020). "Cybersecurity for Industrial Control Systems: A Survey." In: *computers & security* 89, p. 101677.
- Bhatt, Sandeep, Manadhata, Pratyusa K, and Zomlot, Loai (2014). "The Operational Role of Security Information and Event Management Systems." In: *IEEE security & Privacy* 12.5, pp. 35–41.
- Blockley, DI, Agarwal, J, Pinto, JT, and Woodman, NJ (2002). "Structural vulnerability, reliability and risk." In: *Progress in Structural Engineering and Materials* 4.2, pp. 203–212.
- Bodenheim, Roland, Butts, Jonathan, Dunlap, Stephen, and Mullins, Barry (2014). "Evaluation of the Ability of the Shodan Search Engine to Identify Internet-Facing Industrial Control Devices." In: *International Journal of Critical Infrastructure Protection* 7.2, pp. 114–123.
- Bordel, Borja, Alcarria, Ramón, Robles, Tomás, and Martín, Diego (2017). "Cyber-physical systems: Extending pervasive sensing from control theory to the Internet of Things." In: *Pervasive and mobile computing* 40, pp. 156–184.
- Boyer, Stuart A (2009). *SCADA: supervisory control and data acquisition*. International Society of Automation.
- Boyes, Hugh, Hallaq, Bil, Cunningham, Joe, and Watson, Tim (2018). "The industrial internet of things (IIoT): An analysis framework." In: *Computers in industry* 101, pp. 1–12.
- Bozorgi, Mehran, Saul, Lawrence K, Savage, Stefan, and Voelker, Geoffrey M (2010). "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.
- Brand, Klaus-Peter, Wimmer, Wolfgang, and Lohmann, Volker (2003). *Substation Automation Handbook*. Utility Automation Consulting Lohmann Bremgarten, Switzerland.
- Brand, KP, Brunner, Ch, and Wimmer, Wolfgang (2011). "Design of IEC 61850 Based Substation Automation Systems According to Customer Requirements." In: *Indian Journal of Power and River Valley Development* 61.5, p. 87.
- Brazhuk, Andrei (2019). "Semantic Model of Attacks and Vulnerabilities based on CAPEC and CWE Dictionaries." In: *International Journal of Open Information Technologies* 7.3, pp. 38–41.
- Breiman, Leo (1996). "Bagging Predictors." In: *Machine learning* 24.2, pp. 123–140.
- Brown, Sarah, Gommers, Joep, and Serrano, Oscar (2015). "From Cyber Security Information Sharing to Threat Management." In: *Proceedings of the 2nd ACM workshop on information sharing and collaborative security*, pp. 43–49.
- Buczak, Anna L and Guven, Erhan (2015). "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection." In: *IEEE Communications surveys & tutorials* 18.2, pp. 1153–1176.

- Bullough, Benjamin L, Yanchenko, Anna K, Smith, Christopher L, and Zipkin, Joseph R (2017). “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.
- Burgess, J Peter (2010). *Handbook of New Security Studies*. Routledge.
- Byers, Robert, Waltermire, David, Turner, Christopher, et al. (2020). *Collaborative Vulnerability Metadata Acceptance Process (CVMAP) for CVE Numbering Authorities (CNAs) and Authorized Data Publishers*. Tech. rep. National Institute of Standards and Technology.
- Bytschkow, Denis, Campetelli, Alarico, Cengarle, Mar ía Victoria, Irlbeck, Maximilian, and Schorp, Konstantin (2014). “Reference Framework for the Engineering of Cyber-Physical Systems: A First Approach.” In.
- CAPEC View With Entry ID 658 (2022). <https://capec.mitre.org/data/definitions/658.html>. Accessed: 2022-04-30.
- Cardenas, Alvaro, Amin, Saurabh, Sinopoli, Bruno, Giani, Annarita, Perrig, Adrian, Sastry, Shankar, et al. (2009). “Challenges for securing cyber physical systems.” In: *Workshop on future directions in cyber-physical systems security*. Vol. 5. 1. Citeseer.
- Cárdenas, Alvaro A, Amin, Saurabh, Lin, Zong-Syun, Huang, Yu-Lun, Huang, Chi-Yen, and Sastry, Shankar (2011). “Attacks Against Process Control Systems: Risk Assessment, Detection, and Response.” In: *Proceedings of the 6th ACM symposium on information, computer and communications security*, pp. 355–366.
- CERT Alerts (2022). <https://us-cert.cisa.gov/ncas/alerts>. Accessed: 2022-04-23.
- CERT Coordination Center (CC) Vulnerability Note Database (VND) (2022). <https://www.kb.cert.org/vuls/>. Accessed: 2022-04-23.
- CERT Coordination Center Vulnerability Data Archive (2020). <https://github.com/CERTCC/Vulnerability-Data-Archive>. Accessed: 2022-04-23.
- CERT Vulnerability Notes (2020). <https://vuls.cert.org/confluence/display/tools/CERT+Vulnerability+Data+Archive+and+Tools>. Accessed: 2022-04-23.
- Chan, Leong, Morgan, Ian, Simon, Hayden, Alshabanat, Fares, Ober, Devin, Gentry, James, Min, David, and Cao, Renzhi (2019). “Survey of AI in Cybersecurity for Information Technology Management.” In: *2019 IEEE Technology & Engineering Management Conference (TEMSCON)*. IEEE, pp. 1–8.
- Chang, Fredrick R. (2012). *Guest Editor’s Column. The Next Wave*, 19(4): 1–2. url: <https://www.nsa.gov/Portals/70/documents/resources/everyone/digital-media-center/publications/the-next-wave/TNW-19-4.pdf>.
- Chang, Yung-Yu, Zavarsky, Pavol, Ruhl, Ron, and Lindskog, Dale (2011). “Trend Analysis of The CVE for Software Vulnerability Management.” In: *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. IEEE, pp. 1290–1293.

- Chaparro, Oscar, Lu, Jing, Zampetti, Fiorella, Moreno, Laura, Di Penta, Massimiliano, Marcus, Andrian, Bavota, Gabriele, and Ng, Vincent (2017). “Detecting Missing Information in Bug Descriptions.” In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pp. 396–407.
- Chen, Haipeng, Liu, Rui, Park, Noseong, and Subrahmanian, VS (2019). “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.
- Chen, Huashan, Cho, Jin-Hee, and Xu, Shouhuai (2018). “Quantifying the Security Effectiveness of Network Diversity: Poster.” In: *Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security*, pp. 1–1.
- Chen, Ping, Desmet, Lieven, and Huygens, Christophe (2014). “A Study on Advanced Persistent Threats.” In: *IFIP International Conference on Communications and Multimedia Security*. Springer, pp. 63–72.
- Chen, Tse-Hsun, Thomas, Stephen W, and Hassan, Ahmed E (2016). “A Survey on the Use of Topic Models When Mining Software Repositories.” In: *Empirical Software Engineering* 21.5, pp. 1843–1919.
- Cho, Ki-Seon, Shin, Joong-Rin, and Hyun, Seung Ho (2001). “Optimal Placement of Phasor Measurement Units with GPS Receiver.” In: *2001 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No. 01CH37194)*. Vol. 1. IEEE, pp. 258–262.
- Chopade, Pravin and Bikdash, Marwan (2011). “Critical infrastructure interdependency modeling: Using graph models to assess the vulnerability of smart power grid and SCADA networks.” In: *2011 8th International Conference & Expo on Emerging Technologies for a Smarter World*. IEEE, pp. 1–6.
- Christey, Steve and Martin, Brian (2013). “Buying into the Bias: Why Vulnerability Statistics Suck.” In: *BlackHat, Las Vegas, USA, Tech. Rep* 1.
- Christey, Steve and Martin, Robert A (2007). *Vulnerability Type Distributions in CVE*.
- Ciapessoni, Emanuele, Cirio, Diego, Kjølle, Gerd, Massucco, Stefano, Pitto, Andrea, and Sforza, Marino (2016). “Probabilistic Risk-Based Security Assessment of Power Systems Considering Incumbent Threats and Uncertainties.” In: *IEEE Transactions on Smart Grid* 7.6, pp. 2890–2903.
- Cloutier, Robert, Muller, Gerrit, Verma, Dinesh, Nilchiani, Roshanak, Hole, Eirik, and Bone, Mary (2010). “The concept of reference architectures.” In: *Systems Engineering* 13.1, pp. 14–27.
- Collberg, Christian and Proebsting, Todd A (2016). “Repeatability in Computer Systems Research.” In: *Communications of the ACM* 59.3, pp. 62–69.
- Common Attack Pattern Enumeration and Classification (CAPEC)* (2022). <https://capec.mitre.org/index.html>. Accessed: 2022-04-30.

- Common Configuration Enumeration (CCE)* (2022). <https://cce.mitre.org/>. Accessed: 2022-04-30.
- Common Platform Enumeration (CPE)* (2022). <https://cpe.mitre.org/>. Accessed: 2022-04-23.
- Common Vulnerability Enumeration (CVE)* (2022). <https://www.cve.org/>. Accessed: 2022-04-30.
- Common Vulnerability Reporting Framework* (2021). <https://www.first.org/newsroom/releases/20210601>. Accessed: 2022-04-30.
- Common Vulnerability Scoring System (CVSS)* (2022). <https://www.first.org/cvss/>. Accessed: 2022-04-30.
- Common Weakness Enumeration (CWE)* (2022). <https://cwe.mitre.org/index.html>. Accessed: 2022-04-30.
- Computer Aided Integration of Requirements and Information Security* (2022). <https://cairis.org/>. Accessed: 2022-04-30.
- Computer Emergency Response Team* (2022). <https://www.cisa.gov/uscert>. Accessed: 2022-04-23.
- Conklin, Wm Arthur (2016). “IT vs. OT security: A time to consider a change in CIA to include resilience.” In: *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, pp. 2642–2647.
- Corporation, MITRE (Nov. 2010). *Science of Cyber-Security*. url: <https://fas.org/irp/agency/dod/jason/cyber.pdf>.
- Corporation, North American Electric Reliability (2014). *NERC CIP Standards, CIP Version 5*. url: <https://www.nerc.com/pa/CI/Pages/Transition-Program.aspx>.
- CountVectorizer* (2022). [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_extraction.text.CountVectorizer.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html). Accessed: 2022-04-23.
- Craigen, Dan, Diakun-Thibault, Nadia, and Purse, Randy (2014). “Defining Cybersecurity.” In: *Technology Innovation Management Review* 4.10.
- Creswell, John W and Clark, Vicki L Plano (2017). *Designing and Conducting Mixed Methods Research*. Sage publications.
- Critical Infrastructure Sectors* (2022). <https://www.cisa.gov/critical-infrastructure-sectors>. Accessed: 2022-04-23.
- Croft, Roland, Xie, Yongzheng, and Babar, Muhammad Ali (2022). “Data preparation for software vulnerability prediction: A systematic literature review.” In: *IEEE Transactions on Software Engineering*.
- CVE Automation Working Group* (2022). <https://github.com/CVEProject/automation-working-group>. Accessed: 2022-04-30.

- CVE Details* (2022). <https://www.cvedetails.com/vulnerabilities-by-types.php>. Accessed: 2022-04-30.
- CVE Numbering Authorities* (2022). <https://cve.mitre.org/cve/cna.html>. Accessed: 2022-04-30.
- CVE-2020-0964* (2020). <https://nvd.nist.gov/vuln/detail/CVE-2020-0964>. Accessed: 2022-04-30.
- CVE-2020-0966* (2021). <https://nvd.nist.gov/vuln/detail/CVE-2020-0966>. Accessed: 2022-04-30.
- CVE-2020-15782* (2020). <https://nvd.nist.gov/vuln/detail/CVE-2020-15782>. Accessed: 2022-04-30.
- CVE-2021-1361* (2021). <https://nvd.nist.gov/vuln/detail/CVE-2021-1361>. Accessed: 2022-04-30.
- CVE-2021-34523* (2021). <https://nvd.nist.gov/vuln/detail/CVE-2021-34523>. Accessed: 2022-04-30.
- CVE-2021-37172* (2021). <https://nvd.nist.gov/vuln/detail/CVE-2021-37172>. Accessed: 2022-04-30.
- CVE-2021-44228* (2021). <https://nvd.nist.gov/vuln/detail/CVE-2021-44228>. Accessed: 2022-04-30.
- CVE-2021-45046* (2021). <https://nvd.nist.gov/vuln/detail/CVE-2021-45046>. Accessed: 2022-04-30.
- CVE-2021-45105* (2021). <https://nvd.nist.gov/vuln/detail/CVE-2021-45105>. Accessed: 2022-04-30.
- CVE-Search* (2022). <https://www.circl.lu/services/cve-search/>. Accessed: 2022-04-23.
- CVEs and the NVD Process* (2022). <https://nvd.nist.gov/general/cve-process>. Accessed: 2022-04-23.
- CVSS Documentation* (2022). <https://www.first.org/cvss/specification-document>. Accessed: 2022-04-30.
- CVSS V2 Documentation* (2022). <https://www.first.org/cvss/v2/guide>. Accessed: 2022-04-30.
- CVSS V3 Documentation* (2022). <https://www.first.org/cvss/v3.1/specification-document>. Accessed: 2022-04-30.
- CWE-119* (2022). <https://cwe.mitre.org/data/definitions/119.html>. Accessed: 2022-04-30.
- Cyber-Physical Systems - A Concept Map* (2022). <https://ptolemy.berkeley.edu/projects/cps/>. Accessed: 2022-04-30.
- Debole, Franca and Sebastiani, Fabrizio (2004). “Supervised Term Weighting for Automated Text Categorization.” In: *Text Mining and Its Applications*. Springer, pp. 81–97.

- Degabriele, Jean Paul, Paterson, Kenny, and Watson, Gaven (2010). "Provable Security in the Real World." In: *IEEE Security & Privacy* 9.3, pp. 33–41.
- Delen, Dursun and Demirkhan, Haluk (2013). *Data, Information and Analytics as Services*.
- Den Braber, Folker, Hogganvik, Ida, Lund, M Soldal, Stølen, Ketik, and Vraalsen, Fredrik (2007). "Model-Based Security Analysis in Seven Steps - Guided Tour to the CORAS Method." In: *BT Technology Journal* 25.1, pp. 101–117.
- Dietterich, Thomas G (2000). "Ensemble Methods in Machine Learning." In: *International workshop on multiple classifier systems*. Springer, pp. 1–15.
- Dilek, Selma, Çakır, Hüseyin, and Aydin, Mustafa (2015). "Applications of Artificial Intelligence Techniques to Combating Cyber Crimes: A Review." In: *arXiv preprint arXiv:1502.03552*.
- Disterer, Georg (2013). "ISO/IEC 27000, 27001 and 27002 for information security management." In.
- Dittrich, David, Kenneally, Erin, et al. (2012). *The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research*. Tech. rep. US Department of Homeland Security.
- Docker (2022). <https://www.docker.com/products>. Accessed: 2022-04-23.
- Dong, Ying, Guo, Wenbo, Chen, Yueqi, Xing, Xinyu, Zhang, Yuqing, and Wang, Gang (2019). "Towards the Detection of Inconsistencies in Public Security Vulnerability Reports." In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pp. 869–885.
- Doynikova, Elena, Fedorchenko, Andrey, and Kotenko, Igor (2019). "Ontology of Metrics for Cyber Security Assessment." In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pp. 1–8.
- Drag&Bot (2022). <https://www.dragandbot.com/>. Accessed: 2022-04-23.
- Duque-Ramos, Astrid, Boeker, Martin, Jansen, Ludger, Schulz, Stefan, Iniesta, Miguela, and Fernández-Breis, Jesualdo Tomás (2014). "Evaluating the Good Ontology Design Guideline (GoodOD) with the Ontology Quality Requirements and Evaluation Method and Metrics (OQuaRE)." In: *PloS one* 9.8, e104463.
- Eckhart, Matthias and Ekelhart, Andreas (2018). "Towards Security-Aware Virtual Environments for Digital Twins." In: *Proceedings of the 4th ACM workshop on cyber-physical system security*, pp. 61–72.
- Edkrantz, Michel and Said, Alan (2015). "Predicting Cyber Vulnerability Exploits with Machine Learning." In: *SCAI*, pp. 48–57.
- EKANS Ransomware and ICS Operations (2020). <https://www.dragos.com/blog/industry-news/ekans-ransomware-and-ics-operations/>. Accessed: 2022-04-30.

- Ekelhart, Andreas, Fenz, Stefan, and Neubauer, Thomas (2009). "Aurum: A Framework for Information Security Risk Management." In: *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*. IEEE, pp. 1–10.
- Ellerm, Augustus and Morales-Trujillo, Miguel Ehécatl (2020). "Modelling Security Aspects with ArchiMate: A Systematic Mapping Study." In: *46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, Portoroz, Slovenia, August 26–28, 2020*. IEEE, pp. 577–584.
- Eusgeld, Irene, Nan, Cen, and Dietz, Sven (2011). "“System-of-Systems” Approach for Interdependent Critical Infrastructures." In: *Reliability Engineering & System Safety* 96.6, pp. 679–686.
- Evans, David and Stolfo, Sal (2011). "Guest Editors' Introduction: The Science of Security." In: *IEEE Security & Privacy* 9.3, pp. 16–17.
- Exploit Database* (2022). <https://github.com/offensive-security/exploitdb>. Accessed: 2022-04-23.
- Ezell, Barry Charles (2007). "Infrastructure Vulnerability Assessment Model (I-VAM)." In: *Risk Analysis: An International Journal* 27.3, pp. 571–583.
- Fagroud, Fatima Zahra, Ajallouda, Lahbib, Toumi, Hicham, Achtaich, Khadija, El Filali, Sanaa, et al. (2020). "IOT Search Engines: Exploratory Data Analysis." In: *Procedia Computer Science* 175, pp. 572–577.
- Falliere, Nicolas, Murchu, Liam O, and Chien, Eric (2011). "W32. Stuxnet Dossier." In: *White Paper, Symantec Corp., Security Response* 5.6, p. 29.
- Fang, Xi, Misra, Satyajayant, Xue, Guoliang, and Yang, Dejun (2011). "Smart Grid - The New and Improved Power Grid: A Survey." In: *IEEE Communications Surveys & Tutorials* 14.4, pp. 944–980.
- Fang, Yong, Liu, Yongcheng, Huang, Cheng, and Liu, Liang (2020). "FastEmbed: Predicting Vulnerability Exploitation Possibility Based on Ensemble Machine Learning Algorithm." In: *Plos one* 15.2, e0228439.
- Farhang, Sadegh, Kirdan, Mehmet Bahadir, Laszka, Aron, and Grossklags, Jens (2020). "An empirical study of Android security bulletins in different vendors." In: *Proceedings of The Web Conference 2020*, pp. 3063–3069.
- Farquhar, Bill (1991). "One Approach to Risk Assessment." In: *Computers & Security* 10.1, pp. 21–23.
- Farris, Katheryn A, Shah, Ankit, Cybenko, George, Ganesan, Rajesh, and Jajodia, Sushil (2018). "Vulcon: A System for Vulnerability Prioritization, Mitigation, and Management." In: *ACM Transactions on Privacy and Security (TOPS)* 21.4, pp. 1–28.
- Feiler, Peter H, Lewis, Bruce, and Vestal, Steve (2003). *The SAE avionics architecture description language (AADL) standard: A basis for model-based architecture-driven embedded systems engineering*. Tech. rep. Army Aviation and Missile Command RedStone Arsenal AL.

- Feng, Charles, Wu, Shuning, and Liu, Ningwei (2017). “A User-Centric Machine Learning Framework for Cyber Security Operations Centre.” In: *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, pp. 173–175.
- Flick, Uwe (2015). *Introducing Research Methodology: A Beginner’s Guide to Doing A Research Project*. Sage.
- Fredriksen, Rune, Kristiansen, Monica, Gran, Bjørn Axel, Stølen, Ketil, Opperud, Tom Arthur, and Dimitrakos, Theo (2002). “The CORAS framework for a model-based risk management process.” In: *International Conference on Computer Safety, Reliability, and Security*. Springer, pp. 94–105.
- Frei, Stefan, May, Martin, Fiedler, Ulrich, and Plattner, Bernhard (2006). “Large-Scale Vulnerability Analysis.” In: *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense*. ACM, pp. 131–138.
- Freund, Yoav and Schapire, Robert E (1995). “A Desicion-Theoretic Generalization of Online Learning and An Application to Boosting.” In: *European conference on computational learning theory*. Springer, pp. 23–37.
- Fruhwirth, Christian and Mannisto, Tomi (2009). “Improving CVSS-based Vulnerability Prioritization and Response with Context Information.” In: *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, pp. 535–544.
- FuzzyWuzzy: Fuzzy String Matching in Python* (2021). <https://github.com/seatgeek/fuzzywuzzy>. Accessed: 2022-04-23.
- Gawron, Marian, Cheng, Feng, and Meinel, Christoph (2017). “Automatic Vulnerability Classification Using Machine Learning.” In: *International Conference on Risks and Security of Internet and Systems*. Springer, pp. 3–17.
- Geer, Dan and Roytman, Michael (2013). “Measuring vs. Modelling.” In: *; login:: the Magazine of USENIX & SAGE* 38.6, pp. 64–67.
- Ghaffarian, Seyed Mohammad and Shahriari, Hamid Reza (2017). “Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey.” In: *ACM Computing Surveys (CSUR)* 50.4, p. 56.
- Giraldo, Jairo, Sarkar, Esha, Cardenas, Alvaro A, Maniatakos, Michail, and Kantarcio glu, Murat (2017). “Security and Privacy in Cyber-Physical Systems: A Survey of Surveys.” In: *IEEE Design & Test* 34.4, pp. 7–17.
- Goerlandt, Floris and Reniers, Genserik (2016). “On the Assessment of Uncertainty in Risk Diagrams.” In: *Safety Science* 84, pp. 67–77.
- González-Barahona, Jesús M and Robles, Gregorio (2012). “On the reproducibility of empirical software engineering studies based on data retrieved from development repositories.” In: *Empirical Software Engineering* 17.1, pp. 75–89.
- Gottschalk, Marion, Uslar, Mathias, and Delfs, Christina (2017). *The Use Case and Smart Grid Architecture Model Approach: The IEC 62559-2 Use Case Template and the SGAM Applied in Various Domains*. Springer.

- Grandry, Eric, Feltus, Christophe, and Dubois, Eric (2013). "Conceptual Integration of Enterprise Architecture Management and Security Risk Management." In: *17th IEEE International Enterprise Distributed Object Computing Conference Workshops, EDOC Workshops, Vancouver, BC, Canada, September 9-13, 2013*. Ed. by Ebrahim Bagheri, Dragan Gasevic, Sylvain Hallé, Marek Hatala, Hamid R. Motahari Nezhad, and Manfred Reichert. IEEE Computer Society, pp. 114–123.
- Gregor, Shirley (2006). "The nature of theory in information systems." In: *MIS quarterly*, pp. 611–642.
- Griffor, Edward R, Greer, Christopher, Wollman, David A, and Burns, Martin J (2017). *Framework for Cyber-Physical Systems: Volume 1, Overview*. Tech. rep.
- Guo, Hengdao, Zheng, Ciyan, Iu, Herbert Ho-Ching, and Fernando, Tyrone (2017). "A critical review of cascading failure analysis and modeling of power system." In: *Renewable and Sustainable Energy Reviews* 80, pp. 9–22.
- Hafiz, Munawar and Fang, Ming (2016). "Game of Detections: How Are Security Vulnerabilities Discovered in the Wild?" In: *Empirical Software Engineering* 21.5, pp. 1920–1959.
- Haldar, Rishin and Mukhopadhyay, Debajyoti (2011). "Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach." In: *arXiv preprint arXiv:1101.1232*.
- Han, Zhuobing, Li, Xiaohong, Liu, Hongtao, Xing, Zhenchang, and Feng, Zhiyong (2018). "Deepweak: Reasoning Common Software Weaknesses via Knowledge Graph Embedding." In: *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, pp. 456–466.
- He, Haibo and Yan, Jun (2016). "Cyber-Physical Attacks and Defences in the Smart Grid: a Survey." In: *IET Cyber-Physical Systems: Theory & Applications* 1.1, pp. 13–27.
- Heale, Roberta and Twycross, Alison (2015). "Validity and reliability in quantitative studies." In: *Evidence-based nursing* 18.3, pp. 66–67.
- Heelan, Sean (2011). "Vulnerability Detection Systems: Think Cyborg, Not Robot." In: *IEEE Security & Privacy* 9.3, pp. 74–77.
- Hendrickx, Julien M, Johansson, Karl Henrik, Jungers, Raphael M, Sandberg, Henrik, and Sou, Kin Cheong (2014). "Efficient Computations of a Security Index for False Data Attacks in Power Networks." In: *IEEE Transactions on Automatic Control* 59.12, pp. 3194–3208.
- Herley, Cormac and Van Oorschot, Paul C (2017). "Sok: Science, Security and the Elusive Goal of Security as a Scientific Pursuit." In: *2017 IEEE symposium on security and privacy (SP)*. IEEE, pp. 99–120.
- Hevner, Alan and Chatterjee, Samir (2010). "Design science research in information systems." In: *Design research in information systems*. Springer, pp. 9–22.
- Hevner, Alan, March, Salvatore T, Park, Jinsoo, Ram, Sudha, et al. (2004). "Design Science Research in Information Systems." In: *MIS quarterly* 28.1, pp. 75–105.

- Hevner, Alan R (2007). "A three cycle view of design science research." In: *Scandinavian journal of information systems* 19.2, p. 4.
- Holm, Hannes, Karresand, Martin, Vidström, Arne, and Westring, Erik (2015). "A Survey of Industrial Control System Testbeds." In: *Nordic Conference on Secure IT Systems*. Springer, pp. 11–26.
- Holm, Hannes, Sommestad, Teodor, Almroth, Jonas, and Persson, Mats (2011). "A Quantitative Evaluation of Vulnerability Scanning." In: *Information Management & Computer Security* 19.4, pp. 231–247.
- Homeland Security, U.S. Department of (July 2020). *Critical Infrastructure Sectors*. url: <https://www.cisa.gov/critical-infrastructure-sectors>.
- Hong, Jin B, Kim, Dong Seong, and Haqiq, Abdelkrim (2014). "What vulnerability do we need to patch first?" In: *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, pp. 684–689.
- Houmb, Siv Hilde, Franqueira, Virginia NL, and Engum, Erlend A (2010). "Quantifying Security Risk Level from CVSS Estimates of Frequency and Impact." In: *Journal of Systems and Software* 83.9, pp. 1622–1634.
- Householder, Allen D, Wassermann, Garret, Manion, Art, and King, Chris (2017). *The CERT Guide to Coordinated Vulnerability Disclosure*. Tech. rep. Carnegie-Mellon Univ Pittsburgh Pa Pittsburgh United States.
- Huang, Guoyan, Li, Yazhou, Wang, Qian, Ren, Jiadong, Cheng, Yongqiang, and Zhao, Xiaolin (2019). "Automatic Classification Method for Software Vulnerability Based on Deep Neural Network." In: *IEEE Access* 7, pp. 28291–28298.
- Huang, Shuguang, Tang, Heping, Zhang, Min, and Tian, Jie (2010). "Text Clustering on National Vulnerability Database." In: *2010 Second International Conference on Computer Engineering and Applications*. Vol. 2. IEEE, pp. 295–299.
- Humayed, Abdulmalik, Lin, Jingqiang, Li, Fengjun, and Luo, Bo (2017). "Cyber-Physical Systems Security - A Survey." In: *IEEE Internet of Things Journal* 4.6, pp. 1802–1831.
- Husari, Ghaith, Niu, Xi, Chu, Bill, and Al-Shaer, Ehab (2018). "Using Entropy and Mutual Information to Extract Threat Actions from Cyber Threat Intelligence." In: *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, pp. 1–6.
- Husari, Ghaith, Al-Shaer, Ehab, Ahmed, Mohiuddin, Chu, Bill, and Niu, Xi (2017). "Ttpdrill: Automatic and Accurate Extraction of Threat Actions from Unstructured Text of CTI Sources." In: *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, pp. 103–115.
- ICS-CERT Advisories* (2022). <https://us-cert.cisa.gov/ics/advisories>. Accessed: 2022-04-23.
- InsightVM* (2022). <https://www.rapid7.com/products/insightvm/>. Accessed: 2022-04-23.

- IRC5* (2022). <https://new.abb.com/products/robotics/controllers/irc5>. Accessed: 2022-04-23.
- IriusRisk* (2022). <https://iriusrisk.com/>. Accessed: 2022-04-30.
- Irlbeck, Maximilian, Bytschkow, Denis, Hackenberg, Georg, and Koutsoumpas, Vasileios (2013). “Towards a bottom-up development of reference architectures for smart energy systems.” In: *2013 2nd International Workshop on Software Engineering Challenges for the Smart Grid (SE4SG)*. IEEE, pp. 9–16.
- ISA, ANSI (2007). “ISA-99. 00. 01-2007 Security for Industrial Automation and Control Systems Part 1 Terminology, Concepts, and Models.” In: *International Society for Automation* 10.
- Jacobs, Pierre, Arnab, Alapan, and Irwin, Barry (2013). “Classification of Security Operation Centres.” In: *2013 Information Security for South Africa*. IEEE, pp. 1–7.
- Jajodia, Sushil, Noel, Steven, Kalapa, Pramod, Albanese, Massimiliano, and Williams, John (2011). “Cauldron Mission-Centric Cyber Situational Awareness with Defense in Depth.” In: *MILCOM*, pp. 1339–1344.
- Jang-Jaccard, Julian and Nepal, Surya (2014). “A Survey of Emerging Threats in Cybersecurity.” In: *Journal of Computer and System Sciences* 80.5, pp. 973–993.
- Jarke, Matthias, Gallersdörfer, Rainer, Jeusfeld, Manfred A., and Staudt, Martin (1995). “ConceptBase - A Deductive Object Base for Meta Data Management.” In: *J. Intell. Inf. Syst.* 4.2, pp. 167–192.
- Jeusfeld, Manfred, Jarke, Matthias, and Mylopoulos, John (2009). *Metamodeling for Method Engineering*. MIT press Cambridge.
- Jiang, Jie, Yu, Lele, Jiang, Jiawei, Liu, Yuhong, and Cui, Bin (2018a). “Angel: A New Large-Scale Machine Learning System.” In: *National Science Review* 5.2, pp. 216–236.
- Jiang, Yuning (2021). *NVD Feature Analysis*. <https://github.com/Yuning-J/NVDFeatureAnalysis>. Accessed: 2022-04-23.
- Jiang, Yuning and Atif, Yacine (2020). “An Approach to Discover and Assess Vulnerability Severity Automatically in Cyber-Physical Systems.” In: *13th International Conference on Security of Information and Networks*, pp. 1–8.
- Jiang, Yuning and Atif, Yacine (2021). “A Selective Ensemble Model for Cognitive Cybersecurity Analysis.” In: *Journal of Network and Computer Applications* 193, p. 103210.
- Jiang, Yuning and Atif, Yacine (2022). “Towards Automatic Discovery and Assessment of Vulnerability Severity in Cyber-Physical Systems.” In: *Journal of ARRAY (Available at SSRN 4019226)*.
- Jiang, Yuning, Atif, Yacine, and Ding, Jianguo (2019). “Cyber-Physical Systems Security Based on a Cross-Linked and Correlated Vulnerability Database.” In: *International Conference on Critical Information Infrastructures Security*. Springer, pp. 71–82.

- Jiang, Yuning, Atif, Yacine, Ding, Jianguo, and Wang, Wei (2019). “A Semantic Framework with Humans in the Loop for Vulnerability-Assessment in Cyber-Physical Production Systems.” In: *International Conference on Risks and Security of Internet and Systems*. Springer, pp. 128–143.
- Jiang, Yuning, Jeusfeld, Manfred, Atif, Yacine, Ding, Jianguo, Brax, Christoffer, and Nero, Eva (2018b). “A Language and Repository for Cyber Security of Smart Grids.” In: *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, pp. 164–170.
- Jiang, Yuning, Jeusfeld, Manfred, and Ding, Jianguo (2021). “Evaluating the Data Inconsistency of Open-Source Vulnerability Repositories.” In: *The 16th International Conference on Availability, Reliability and Security*, pp. 1–10.
- Jiang, Yuning, Jeusfeld, Manfred, and Ding, Jianguo (2022). “Model-Based Cybersecurity Analysis: Extending Enterprise Modeling to Critical Infrastructure Cybersecurity.” In: *Journal of Business & Information Systems Engineering (Revision Submitted)*.
- Jo, Hyeonseong, Kim, Jinwoo, Porras, Phillip, Yegneswaran, Vinod, and Shin, Seungwon (2020). “GapFinder: Finding Inconsistency of Security Information From Unstructured Text.” In: *IEEE Transactions on Information Forensics and Security* 16, pp. 86–99.
- Joachims, Thorsten (2001). “A Statistical Learning Model of Text Classification for Support Vector Machines.” In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 128–136.
- Joh, HyunChul and Malaiya, Yashwant K (2011). “Defining and Assessing Quantitative Security Risk Measures Using Vulnerability Lifecycle and CVSS Metrics.” In: *The 2011 International Conference on Security and Management (SAM)*, pp. 10–16.
- Johannesson, Paul and Perjons, Erik (2014). *An Introduction to Design Science*. Springer.
- Johnson, Christopher, Badger, Mark, Waltermire, David, Snyder, Julie, and Skorupka, Clem (2016a). *Guide to Cyber Threat Information Sharing*. Tech. rep. National Institute of Standards and Technology.
- Johnson, Pontus, Lagerström, Robert, Ekstedt, Mathias, and Franke, Ulrik (2016b). “Can the Common Vulnerability Scoring System Be Trusted? A Bayesian Analysis.” In: *IEEE Transactions on Dependable and Secure Computing* 15.6, pp. 1002–1015.
- Johnson, Pontus, Vernotte, Alexandre, Gorton, Dan, Ekstedt, Mathias, and Lagerström, Robert (2016c). “Quantitative Information Security Risk Estimation Using Probabilistic Attack Graphs.” In: *International Workshop on Risk Assessment and Risk-driven Testing*. Springer, pp. 37–52.
- Kampanakis, Panos (2014). “Security Automation and Threat Information-Sharing Options.” In: *IEEE Security & Privacy* 12.5, pp. 42–51.

- Kandias, Miltiadis, Mylonas, Alexios, Theoharidou, Marianthi, and Gritzalis, Dimitris (2011). "Exploitation of auctions for outsourcing security-critical projects." In: *2011 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, pp. 646–651.
- Khaitan, Siddhartha Kumar and McCalley, James D (2015). "Design Techniques and Applications of Cyber-Physical Systems: A Survey." In: *IEEE Systems Journal* 9.2, pp. 350–365.
- Khan, Rafiullah, McLaughlin, Kieran, Laverty, David, and Sezer, Sakir (2017). "STRIDE-based threat modeling for cyber-physical systems." In: *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. IEEE, pp. 1–6.
- Khazaei, Atefeh, Ghasemzadeh, Mohammad, and Derhami, Vali (2016). "An Automatic Method for CVSS Score Prediction Using Vulnerabilities Description." In: *Journal of Intelligent & Fuzzy Systems* 30.1, pp. 89–96.
- Kijewski, Piotr and Pawliński, Paweł (2014). "Proactive detection and automated exchange of network security incidents." In: *Abgerufen am 20.*
- Kittler, Josef, Hatef, Mohamad, Duin, Robert PW, and Matas, Jiri (1998). "On Combining Classifiers." In: *IEEE transactions on pattern analysis and machine intelligence* 20.3, pp. 226–239.
- Knapp, Eric D and Langill, Joel Thomas (2014). *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Syngress.
- Knapp, Eric D and Samani, Raj (2013). *Applied cyber security and the smart grid: implementing security controls into the modern power infrastructure*. Newnes.
- Kong, Peng-Yong (2019). "Optimal configuration of interdependence between communication network and power grid." In: *IEEE Transactions on Industrial Informatics* 15.7, pp. 4054–4065.
- König, Sandra, Rass, Stefan, Rainer, Benjamin, and Schauer, Stefan (2019). "Hybrid dependencies between cyber and physical systems." In: *Intelligent Computing-Proceedings of the Computing Conference*. Springer, pp. 550–565.
- Korman, Matus, Lagerström, Robert, Välja, Margus, Ekstedt, Mathias, and Blom, Rikard (2016). "Technology management through architecture reference models: A smart metering case." In: *2016 Portland International Conference on Management of Engineering and Technology (PICMET)*. IEEE, pp. 2338–2350.
- Kotenko, Igor, Fedorchenko, Andrey, Doynikova, Elena, and Chechulin, Andrey (2018). "An Ontology-Based Storage of Security Information." In: *Information Technology and Control* 47.4, pp. 655–667.
- Koubarakis, Manolis, Borgida, Alexander, Constantopoulos, Panos, Doerr, Martin, Jarke, Matthias, Jeusfeld, Manfred A, Mylopoulos, John, and Plexousakis, Dimitris (2021). "A retrospective on Telos as a metamodeling language for requirements engineering." In: *Requirements Engineering* 26.1, pp. 1–23.

- Kowsari, Kamran, Jafari Meimandi, Kiana, Heidarysafa, Mojtaba, Mendum, Sanjana, Barnes, Laura, and Brown, Donald (2019). "Text Classification Algorithms: A Survey." In: *Information* 10.4, p. 150.
- Krassnig, Christian (2011). "European Programme on Critical Infrastructure Protection (EPCIP)." In: *1st international workshop on regional critical infrastructures protection programmes*, pp. 1–16.
- Kröger, Wolfgang and Zio, Enrico (2011). *Vulnerable Systems*. Springer Science & Business Media.
- Kuhn, Thomas S (2012). *The Structure of Scientific Revolutions*. University of Chicago press.
- Kuppa, Aditya, Aouad, Lamine, and Le-Khac, Nhien-An (2021). "Linking CVE's to MITRE ATT&CK Techniques." In: *The 16th International Conference on Availability, Reliability and Security*, pp. 1–12.
- Kure, Halima, Islam, Shareeful, and Razzaque, Mohammad (2018). "An Integrated Cyber Security Risk Management Approach for a Cyber-Physical System." In: *Applied Sciences* 8.6, p. 898.
- Kwasinski, Alexis (2020). "Modeling of Cyber-Physical Intra-Dependencies in Electric Power Grids and Their Effect on Resilience." In: *2020 8th Workshop on Modeling and Simulation of Cyber-Physical Energy Systems*. IEEE, pp. 1–6.
- Ladd, B (2017). "The Race Between Security Professionals and Adversaries." In: *Recorded Future Blog*. Available online in November.
- Lallie, Harjinder Singh, Debattista, Kurt, and Bal, Jay (2018). "An Empirical Evaluation of the Effectiveness of Attack Graphs and Fault Trees in Cyber-Attack Perception." In: *IEEE Transactions on Information Forensics and Security* 13.5, pp. 1110–1122.
- Landwehr, Carl E (2012). "Cybersecurity: From Engineering to Science." In: *Developing a Blueprint for a Science of Cybersecurity* 2.
- Larcher Jr, Celio HN and Barbosa, Helio JC (2019). "Auto-CVE: a Coevolutionary Approach to Evolve Ensembles in Automated Machine Learning." In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 392–400.
- Larose, Daniel T (2015). *Data Mining and Predictive Analytics*. John Wiley & Sons.
- Le, Triet HM, Chen, Huaming, and Babar, M Ali (2021). "A Survey on Data-Driven Software Vulnerability Assessment and Prioritization." In: *arXiv preprint arXiv: 2107.08364*.
- Le, Triet Huynh Minh, Sabir, Bushra, and Babar, Muhammad Ali (2019). "Automated Software Vulnerability Assessment with Concept Drift." In: *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, pp. 371–382.
- Lee, Edward A (2015). "The Past, Present and Future of Cyber-Physical Systems: A Focus on Models." In: *Sensors* 15.3, pp. 4837–4869.

- Lee, Jay, Bagheri, Behrad, and Kao, Hung-An (2015). “A Cyber-Physical Systems Architecture for Industry 4.0-Based Manufacturing Systems.” In: *Manufacturing letters* 3, pp. 18–23.
- Lepenioti, Katerina, Bousdekis, Alexandros, Apostolou, Dimitris, and Mentzas, Gergoris (2020). “Prescriptive Analytics: Literature Review and Research Challenges.” In: *International Journal of Information Management* 50, pp. 57–70.
- Lewis, Ted G (2019). *Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation*. John Wiley & Sons.
- Li, Longjie, Yu, Yang, Bai, Shenshen, Hou, Ying, and Chen, Xiaoyun (2017). “An Effective Two-Step Intrusion Detection Approach Based on Binary Classification and  $k$ -NN.” In: *IEEE Access* 6, pp. 12060–12073.
- Liao, Q. Vera, Gruen, Daniel, and Miller, Sarah (2020). “Questioning the AI: Informing Design Practices for Explainable AI User Experiences.” In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI ’20. Honolulu, HI, USA: Association for Computing Machinery, pp. 1–15. isbn: 9781450367080. doi: 10.1145/3313831.3376590.
- Liao, Xiaojing, Yuan, Kan, Wang, XiaoFeng, Li, Zhou, Xing, Luyi, and Beyah, Raheem (2016). “Acing the IOC Game: Toward Automatic Discovery and Analysis of Open-Source Cyber Threat Intelligence.” In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’16. Vienna, Austria: ACM, pp. 755–766. isbn: 978-1-4503-4139-4. doi: 10.1145/2976749.2978315.
- Lin, Zongzong, Lu, Wenlian, and Xu, Shouhuai (2019). “Unified Preventive and Reactive Cyber Defence Dynamics is Still Globally Convergent.” In: *IEEE/ACM Transactions on Networking* 27.3, pp. 1098–1111.
- Lindström, Madelene and Olsson, Stefan (2009). “The European programme for critical infrastructure protection.” In: *Crisis management in the European Union*. Springer, pp. 37–59.
- Liska, Allen (2019). “Early Findings: Review of State and Local Government Ransomware Attacks.” In: *Recorded Future* 10.
- Liu, Lin, Eric, SK, and Mylopoulos, John (2009). “Secure-I\*: Engineering Secure Software Systems through Social Analysis.” In: *Int. J. Software and Informatics* 3.1, pp. 89–120.
- Loshin, David (2010). *Master Data Management*. Morgan Kaufmann.
- Lower, Nicholas and Zhan, Felix (2020). “A Study of Ensemble Methods for Cyber Security.” In: *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, pp. 1001–1009.
- Luján-Mora, Sergio, Vassiliadis, Panos, and Trujillo, Juan (2004). “Data Mapping Diagrams for Data Warehouse Design with UML.” In: *International Conference on Conceptual Modeling*. Springer, pp. 191–204.

- Lun, Yuriy Zacchia, D’Innocenzo, Alessandro, Malavolta, Ivano, and Di Benedetto, Maria Domenica (2016). “Cyber-Physical Systems Security: A Systematic Mapping Study.” In: *arXiv preprint arXiv:1605.09641*.
- Madnick, Stuart, Li, Xitong, and Choucri, Nazli (2009). *Experiences and challenges with using CERT data to analyze international cyber security*.
- Mahto, Deepak Kumar and Singh, Lisha (2016). “A Dive into Web Scraper World.” In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, pp. 689–693.
- Marashi, Koosha, Sarvestani, Sahra Sedigh, and Hurson, Ali R (2017). “Consideration of cyber-physical interdependencies in reliability modeling of smart grids.” In: *IEEE Transactions on Sustainable Computing* 3.2, pp. 73–83.
- Martins, Beatriz F, Serrano, Lenin, Reyes, José F, Panach, José Ignacio, Pastor, Oscar, and Rochwerger, Benny (2020). “Conceptual Characterization of Cybersecurity Ontologies.” In: *IFIP Working Conference on The Practice of Enterprise Modeling*. Springer, pp. 323–338.
- Maughan, Douglas, Balenson, David, Lindqvist, Ulf, and Tudor, Zachary (2013). “Crossing the” Valley of Death”: Transitioning Cybersecurity Research into Practice.” In: *IEEE Security & Privacy* 11.2, pp. 14–23.
- Mavroeidis, Vasileios and Bromander, Siri (2017). “Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence.” In: *Intelligence and Security Informatics Conference (EISIC), 2017 European*. IEEE, pp. 91–98.
- Maxion, Roy (2011). “Making Experiments Dependable.” In: *Dependable and Historic Computing*. Springer, pp. 344–357.
- MELSEC-Q PLC (2022). <https://www.mitsubishielectric.com/fa/products/cnt/plcq/items/index.html>. Accessed: 2022-04-23.
- Metasploit (2022). <https://www.metasploit.com/>. Accessed: 2022-04-23.
- Meushaw, Robert (Oct. 2012). *What is security science?* url: <https://cps-vo.org/node/6041>.
- Microsoft Security Response Center (MSRC) (2022). <https://msrc.microsoft.com/update-guide/vulnerability>. Accessed: 2022-04-23.
- Mohurle, Savita and Patil, Manisha (2017). “A Brief Study of WannaCry Threat: Ransomware Attack 2017.” In: *International Journal of Advanced Research in Computer Science* 8.5.
- Mouratidis, Haralambos and Giorgini, Paolo (2007). “Secure Tropos: A Security-Oriented Extension of the Tropos Methodology.” In: *International Journal of Software Engineering and Knowledge Engineering* 17.02, pp. 285–309.
- Mozzaquattro, Bruno A, Melo, Raquel, Agostinho, Carlos, and Jardim-Goncalves, Ricardo (2016). “An Ontology-Based Security Framework for Decision-Making in Industrial Systems.” In: *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. IEEE, pp. 779–788.

- Mozzaquattro, Bruno Augusti, Agostinho, Carlos, Goncalves, Diogo, Martins, João, and Jardim-Goncalves, Ricardo (2018). “An Ontology-Based Cybersecurity Framework for the Internet of Things.” In: *Sensors* 18.9, p. 3053.
- MSRC Security Updates API* (2022). <https://github.com/microsoft/MSRC-Microsoft-Security-Updates-API>. Accessed: 2022-04-23.
- Murray, Glenn, Johnstone, Michael N, and Valli, Craig (2017). “The convergence of IT and OT in critical infrastructure.” In.
- Murtaza, Syed Shariyar, Khreich, Wael, Hamou-Lhadj, Abdelwahab, and Bener, Ayse Basar (2016). “Mining Trends and Patterns of Software Vulnerabilities.” In: *Journal of Systems and Software* 117, pp. 218–228.
- Myhre, Stine Fleischer, Fosso, Olav Bjarte, Heegaard, Poul Einar, Gjerde, Oddbjørn, and Kjølle, Gerd Hovin (2020). “Modeling Interdependencies with Complex Network Theory in a Combined Electrical Power and ICT System.” In: *2020 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*. IEEE, pp. 1–6.
- Mylopoulos, John, Borgida, Alex, Jarke, Matthias, and Koubarakis, Manolis (1990). “Telos: Representing knowledge about information systems.” In: *ACM Transactions on Information Systems (TOIS)* 8.4, pp. 325–362.
- Na, Sarang, Kim, Taeeun, and Kim, Hwankuk (2016). “A Study on the Classification of Common Vulnerabilities and Exposures Using Naive Bayes.” In: *International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer, pp. 657–662.
- Nappa, Antonio, Johnson, Richard, Bilge, Leyla, Caballero, Juan, and Dumitras, Tudor (2015). “The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching.” In: *2015 IEEE symposium on security and privacy*. IEEE, pp. 692–708.
- National Vulnerability Database (NVD)* (2022). <https://nvd.nist.gov/vuln>. Accessed: 2022-04-23.
- Nayak, Kartik, Marino, Daniel, Efstathopoulos, Petros, and Dumitraş, Tudor (2014). “Some Vulnerabilities are Different Than Others.” In: *International Workshop on Recent Advances in Intrusion Detection*. Springer, pp. 426–446.
- Negi, Rohit, Kumar, Parvin, Ghosh, Shibashis, and Shukla, Sandeep K (2019). “Vulnerability Assessment and Mitigation for Industrial Critical Infrastructures with Cyber Physical Test Bed.” In: *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*. IEEE, pp. 145–152.
- Nessus* (2022). <https://www.tenable.com/products/nessus>. Accessed: 2022-04-23.
- Neuhaus, Stephan and Zimmermann, Thomas (2010). “Security Trend Analysis with CVE Topic Models.” In: *Software Reliability Engineering (ISSRE), 2010 IEEE 21st International Symposium On*. IEEE, pp. 111–120.

- Nguyen, Phu H, Ali, Shaukat, and Yue, Tao (2017). “Model-Based Security Engineering for Cyber-Physical Systems: A Systematic Mapping Study.” In: *Information and Software Technology* 83, pp. 116–135.
- Nguyen, Viet Hung and Massacci, Fabio (2013). “The (un)reliability of NVD Vulnerable Versions Data: An Empirical Experiment on Google Chrome Vulnerabilities.” In: *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, pp. 493–498.
- Nickerson, Robert C, Varshney, Upkar, and Muntermann, Jan (2013). “A method for taxonomy development and its application in information systems.” In: *European Journal of Information Systems* 22.3, pp. 336–359.
- NIST (2014). “Framework for Improving Critical Infrastructure Cybersecurity.” In: Accessed: 2022-04-23. url: <https://doi.org/10.6028/NIST.CSWP.02122014>.
- NIST, SP (2013). *800-53 Rev 4, Security and Privacy Controls for Federal Information Systems and Organization*.
- NIST Cybersecurity Framework Version 1.1* (2014). <https://www.nist.gov/cyberframework/framework>. Accessed: 2022-04-30.
- NIST Special Publication (SP) 800-30, Revision 1, Guide for Conducting Risk Assessments* (2012). <https://www.nist.gov/privacy-framework/nist-sp-800-30>. Accessed: 2022-04-30.
- Noel, S, Harley, E, Tam, KH, Limiero, M, and Share, M (2016). “CyGraph: Graph-Based Analytics and Visualization for Cybersecurity.” In: *Handbook of Statistics*. Vol. 35. Elsevier, pp. 117–167.
- North American Electric Reliability Corporation Critical Infrastructure Protection* (2008). <https://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx>. Accessed: 2022-04-23.
- Northern, Bradley, Burks, Trey, Hatcher, Marlana, Rogers, Michael, and Ulybyshev, Denis (2021). “VERCASM-CPS: Vulnerability Analysis and Cyber Risk Assessment for Cyber-Physical Systems.” In: *Information* 12.10, p. 408.
- NVD CWE Slice* (2021). <https://nvd.nist.gov/vuln/categories>. Accessed: 2022-04-23.
- O’Hare, Jamie, Macfarlane, Rich, and Lo, Owen (2019). “Identifying Vulnerabilities Using Internet-Wide Scanning Data.” In: *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*. IEEE, pp. 1–10.
- Oates, Briony J (2005). *Researching Information Systems and Computing*. Sage.
- Oliva, Gustavo A, Santana, Francisco WS, Gerosa, Marco A, and De Souza, Cleidson RB (2011). “Towards a Classification of Logical Dependencies Origins: A Case Study.” In: *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution*, pp. 31–40.

- Onan, Aytuğ, Korukoğlu, Serdar, and Bulut, Hasan (2016). “A Multiobjective Weighted Voting Ensemble Classifier Based on Differential Evolution Algorithm for Text Sentiment Classification.” In: *Expert Systems with Applications* 62, pp. 1–16.
- Open Vulnerability and Assessment Language (OVAL)* (2022). <https://oval.mitre.org/>. Accessed: 2022-04-23.
- Open Web Application Security Project Top Ten 2021* (2022). <https://owasp.org/Top10/>. Accessed: 2022-04-30.
- openSSL* (2022). <https://www.openssl.org/>. Accessed: 2022-04-23.
- Oprea, Alina, Li, Zhou, Norris, Robin, and Bowers, Kevin (2018). “Made: Security Analytics for Enterprise Threat Detection.” In: *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 124–136.
- Ouyang, Min (2014). “Review on Modelling and Simulation of Interdependent Critical Infrastructure Systems.” In: *Reliability engineering & System safety* 121, pp. 43–60.
- Palm, Jenny (2021). “Exploring Limited Capacity in the Grid: Actors, Problems, and Solutions.” In: *Frontiers in Energy Research* 9, p. 199.
- Pavleska, Tanja, Aranha, Helder, Masi, Massimiliano, Grandry, Eric, and Sellitto, Giovanni Paolo (2019). “Cybersecurity Evaluation of Enterprise Architectures: The e-SENS Case.” In: *The Practice of Enterprise Modeling - 12th IFIP Working Conference, PoEM 2019, Luxembourg, Luxembourg, November 27-29, 2019, Proceedings*. Ed. by Jaap Gordijn, Wided Guédria, and Henderik A. Proper. Vol. 369. Lecture Notes in Business Information Processing. Springer, pp. 226–241.
- Peffers, Ken, Tuunanen, Tuure, Rothenberger, Marcus A, and Chatterjee, Samir (2007). “A Design Science Research Methodology for Information Systems Research.” In: *Journal of Management Information Systems* 24.3, pp. 45–77.
- Pendleton, Marcus, Garcia-Lebron, Richard, Cho, Jin-Hee, and Xu, Shouhuai (2016). “A Survey on Systems Security Metrics.” In: *ACM Computing Surveys (CSUR)* 49.4, pp. 1–35.
- PES, I (2008). “Ieee standard for scada and automation systems.” In: *vol. IEEE Std C 37*.
- Philosophy, Stanford Encyclopedia of (2013). *Computer Simulations in Science*. url: <https://seop.illc.uva.nl/entries/simulations-science/>.
- Photovoltaics, Dispersed Generation and Storage, Energy (2011). “IEEE Guide for Smart Grid Interoperability of Energy Technology and Information Technology Operation with the Electric Power System (EPS), End-Use Applications, and Loads.” In: *IEEE Std 2030-2011*, pp. 1–126. doi: 10.1109/IEEESTD.2011.6018239.
- PorterStemmer* (2022). <https://www.nltk.org/howto/stem.html>. Accessed: 2022-04-23.
- Quintal, Kyle, Kantarci, Burak, Erol-Kantarci, Melike, Malton, Andrew, and Walenstein, Andrew (2020). “Enterprise Security with Adaptive Ensemble Learning on

- Cooperation and Interaction Patterns.” In: *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, pp. 1–7.
- Rahm, Erhard and Bernstein, Philip A (2001). “A Survey of Approaches to Automatic Schema Matching.” In: *the VLDB Journal* 10.4, pp. 334–350.
- Rajagopal, Smitha, Kundapur, Poornima Panduranga, and Hareesha, Katiganere Siddaramappa (2020). “A Stacking Ensemble for Network Intrusion Detection Using Heterogeneous Datasets.” In: *Security and Communication Networks* 2020. *Ransomware Impacting Pipeline Operations, Alert (AA20-049A)* (2020). <https://us-cert.cisa.gov/ncas/alerts/aa20-049a>. Accessed: 2022-04-23.
- Resende, Paulo Angelo Alves and Drummond, André Costa (2018). “A Survey of Random Forest Based Methods for Intrusion Detection Systems.” In: *ACM Computing Surveys (CSUR)* 51.3, pp. 1–36.
- Risk Analysis, Society of (2015). *Society of risk analysis, glossary of the specialty group on foundations of risk analysis*. url: <https://www.sra.org/news/sra-develops-glossary-risk-related-terms>.
- RobotStudio* (2022). <https://new.abb.com/products/robotics/robotstudio>. Accessed: 2022-04-23.
- Robson, Colin and McCartan, Kieran (2016). *Real World Research*. John Wiley & Sons.
- Rodriguez, Luis Gustavo Araujo, Trazzi, Julia Selvatici, Fossaluzza, Victor, Campiolo, Rodrigo, and Batista, Daniel Macêdo (2018). “Analysis of Vulnerability Disclosure Delays from the National Vulnerability Database.” In: *Anais do I Workshop de Segurança Cibernética em Dispositivos Conectados*. SBC.
- Ropohl, Günter (1999). “Philosophy of Socio-Technical Systems.” In: *Techné: Research in Philosophy and Technology* 4.3, pp. 186–194.
- Rosas-Casals, Martí, Valverde, Sergi, and Solé, Ricard V (2007). “Topological Vulnerability of the European Power Grid under Errors and Attacks.” In: *International Journal of Bifurcation and Chaos* 17.07, pp. 2465–2475.
- Rosenquist, Matthew (2009). “Prioritizing information security risks with threat agent risk assessment.” In: *Intel Corporation White Paper*.
- Rosenstatter, Thomas and Olovsson, Tomas (2018). “Towards a standardized mapping from automotive security levels to security mechanisms.” In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 1501–1507.
- Rouse, William B. (2003). “Engineering Complex sSystems: Implications for Research in Systems Engineering.” In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 33.2, pp. 154–156.
- Ruan, Keyun (2017). “Introducing Cybernomics: A Unifying Economic Framework for Measuring Cyber Risk.” In: *Computers & Security* 65, pp. 77–89.

- Ruland, Karl Christoph, Sassmannshausen, Jochen, Waedt, Karl, and Zivic, Natasa (2017). "Smart grid security—an overview of standards and guidelines." In: *e & i Elektrotechnik und Informationstechnik* 134.1, pp. 19–25.
- Runeson, Per, Host, Martin, Rainer, Austen, and Regnell, Bjorn (2012). *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.
- Ruohonen, Jukka (2019). "A Look at the Time Delays in CVSS Vulnerability Scoring." In: *Applied Computing and Informatics* 15.2, pp. 129–135.
- Ruohonen, Jukka and Leppänen, Ville (2018). "Toward Validation of Textual Information Retrieval Techniques for Software Weaknesses." In: *International Conference on Database and Expert Systems Applications*. Springer, pp. 265–277.
- Russo, Ernesto Rosario, Di Sorbo, Andrea, Visaggio, Corrado A, and Canfora, Gerardo (2019). "Summarizing Vulnerabilities' Descriptions to Support Experts during Vulnerability Assessment Activities." In: *Journal of Systems and Software* 156, pp. 84–99.
- Ryan, Julie JCH, Mazzuchi, Thomas A, Ryan, Daniel J, De la Cruz, Juliana Lopez, and Cooke, Roger (2012). "Quantifying Information Security Risks Using Expert Judgment Elicitation." In: *Computers & Operations Research* 39.4, pp. 774–784.
- Sabottke, Carl, Suciu, Octavian, and Dumitraş, Tudor (2015). "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.
- Santangelo, Giorgio Valenziano, Colacino, Vincenzo Giuseppe, and Marchetti, Mirco (2021). "Analysis, prevention and detection of ransomware attacks on Industrial Control Systems." In: *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)*. IEEE, pp. 1–5.
- Sarker, Iqbal H, Kayes, ASM, Badsha, Shahriar, Alqahtani, Hamed, Watters, Paul, and Ng, Alex (2020). "Cybersecurity Data Science: an Overview from Machine Learning Perspective." In: *Journal of Big data* 7.1, pp. 1–29.
- Sauerwein, Clemens, Pekaric, Irdin, Felderer, Michael, and Breu, Ruth (2019). "An Analysis and Classification of Public Information Security Data Sources used in Research and Practice." In: *Computers & Security* 82, pp. 140–155.
- Saunders, Mark, Lewis, Philip, and Thornhill, Adrian (2009a). *Research Methods for Business Students*. Pearson education.
- Saunders, Mark, Lewis, Philip, and Thornhill, Adrian (2009b). "Understanding Research Philosophies and Approaches." In: *Research Methods for Business Students* 4.106-135.
- Scandariato, Riccardo, Walden, James, Hovsepyan, Aram, and Joosen, Wouter (2014). "Predicting Vulnerable Software Components via Text Mining." In: *IEEE Transactions on Software Engineering* 40.10, pp. 993–1006.

- Scarfone, Karen and Mell, Peter (2009). "An Analysis of CVSS Version 2 Vulnerability Scoring." In: *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, pp. 516–525.
- Schiffman, Mike (2011). "The common vulnerability reporting framework." In: *An Internet Consortium for Advancement of Security on the Internet (ICASI), Whitepaper, Version 1*.
- Security, Kenna (2018). *Prioritisation to Prediction: Analyzing Vulnerability Remediation Strategies*. url: [http://www.kennasecurity.com/prioritization-to-prediction-report/images/Prioritization%5C\\_to%5C\\_Prediction.pdf](http://www.kennasecurity.com/prioritization-to-prediction-report/images/Prioritization%5C_to%5C_Prediction.pdf).
- SecurityFocus Forum (2021). <https://www.securityfocus.com/>. Accessed: 2022-04-23.
- SEGRID Consortium (2017). *Security for Smart Electricity GRIDs, How to Address the Security Challenges in Smart Grids*. Tech. rep. Segrid.eu. url: <https://seggrid.eu/wp-content/uploads/2017/10/Whitepaper-Segrid-9-FV.pdf>.
- Seni, Giovanni and Elder, John F (2010). "Ensemble Methods in Data Mining: Improving Accuracy through Combining Predictions." In: *Synthesis lectures on data mining and knowledge discovery* 2.1, pp. 1–126.
- Al-Shaer, Rawan, Spring, Jonathan M, and Christou, Eliana (2020). "Learning the Associations of MITRE ATT & CK Adversarial Techniques." In: *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, pp. 1–9.
- Shahzad, Muhammad, Shafiq, Muhammad Zubair, and Liu, Alex X (2012). "A Large Scale Exploratory Analysis of Software Vulnerability Life Cycles." In: *Software Engineering (ICSE), 2012 34th International Conference on*. IEEE, pp. 771–781.
- Shamala, Palaniappan, Ahmad, Rabiah, and Yusoff, Mariana (2013). "A Conceptual Framework of Info Structure for Information Security Risk Assessment (ISRA)." In: *Journal of Information Security and Applications* 18.1, pp. 45–52.
- Sharma, Salii, Velgapudi, Naga Sai, and Pandey, Kamlesh (2017). "Performance analysis of IEEE 9 Bus system using TCSC." In: *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)*. IEEE, pp. 251–256.
- Shepard, Michael (2015). *Getting Started with PowerShell*. Packt Publishing Ltd.
- Shevchenko, Nataliya, Chick, Timothy A, O'Riordan, Paige, Scanlon, Thomas P, and Woody, Carol (2018). *Threat modeling: a summary of available methods*. Tech. rep. Carnegie Mellon University Software Engineering Institute Pittsburgh United States.
- Shodan Database (2022). <https://www.shodan.io/dashboard>. Accessed: 2022-04-23.
- Shostack, Adam (2014). *Threat modeling: Designing for security*. John Wiley & Sons.
- Siboni, Shachar, Sachidananda, Vinay, Meidan, Yair, Bohadana, Michael, Mathov, Yael, Bhairav, Suhas, Shabtai, Asaf, and Elovici, Yuval (2019). "Security Testbed

- for Internet-of-Things Devices.” In: *IEEE Transactions on Reliability* 68.1, pp. 23–44.
- Simon, Herbert A (2019). *The Sciences of the Artificial*. MIT press.
- Smith, Reid G and Eckroth, Joshua (2017). “Building AI Applications: Yesterday, Today, and Tomorrow.” In: *AI Magazine* 38.1, pp. 6–22.
- Sokolova, Marina and Lapalme, Guy (2009). “A Systematic Analysis of Performance Measures for Classification Tasks.” In: *Information processing & management* 45.4, pp. 427–437.
- Somekh, Bridget and Lewin, Cathy (2011). *Theory and Methods in Social Research*. Sage.
- Sommer, Robin and Paxson, Vern (2010). “Outside the Closed World: On Using Machine Learning for Network Intrusion Detection.” In: *2010 IEEE symposium on security and privacy*. IEEE, pp. 305–316.
- Sorower, Mohammad S (2010). “A Literature Survey on Algorithms for Multi-Label Learning.” In: *Oregon State University, Corvallis* 18, pp. 1–25.
- Spanos, Georgios, Angelis, Lefteris, and Toloudis, Dimitrios (2017). “Assessment of Vulnerability Severity Using Text Mining.” In: *Proceedings of the 21st Pan-Hellenic Conference on Informatics*, pp. 1–6.
- Spring, Jonathan M, Moore, Tyler, and Pym, David (2017). “Practicing a Science of Security: a Philosophy of Science Perspective.” In: *Proceedings of the 2017 New Security Paradigms Workshop*, pp. 1–18.
- Standardisation/International Electrotechnical Commissions/Institute of Electrical, ISO/IEC/IEEE (International Organisation for and Engineers), Electronics (2015). “ISO/IEC/IEEE 15288: 2015.” In: *IEEE Standard for Systems and Software Engineering - System Life Cycle Processes*.
- Standardization (ISO), International Organization for (Dec. 2018). *ISO/IEC27000: 2009*.
- Standards, National Institute of and Technologies (Nov. 2016). *National Institute of Standards and Technologies SP 800-160*. url: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160.pdf>.
- Stine, Ian, Rice, Mason, Dunlap, Stephen, and Pecarina, John (2017). “A Cyber Risk Scoring System for Medical Devices.” In: *International Journal of Critical Infrastructure Protection* 19, pp. 32–46.
- Stodden, Victoria C (2010). “Reproducible research: Addressing the need for data and code sharing in computational science.” In: *Computing in Science & Engineering*.
- Stouffer, Keith, Falco, Joe, Scarfone, Karen, et al. (2011). “Guide to industrial control systems (ICS) security.” In: *NIST special publication 800.82*, pp. 16–16.

- Strom, Blake E, Applebaum, Andy, Miller, Doug P, Nickels, Kathryn C, Pennington, Adam G, and Thomas, Cody B (2018). “Mitre att&ck: Design and philosophy.” In: *Technical Report*.
- Sukhija, Nitin, Sevin, Sonny, Bautista, Elizabeth, and Dampier, David (2019). *Prescriptive and Predictive Analytics Techniques for Enabling Cybersecurity*.
- Sun, Jiamou, Xing, Zhenchang, Guo, Hao, Ye, Deheng, Li, Xiaohong, Xu, Xiwei, and Zhu, Liming (2021). “Generating Informative CVE Description From ExploitDB Posts by Extractive Summarization.” In: *arXiv preprint arXiv:2101.01431*.
- Sundaramurthy, Sathya Chandran, Bardas, Alexandru G, Case, Jacob, Ou, Xinming, Wesch, Michael, McHugh, John, and Rajagopalan, S Raj (2015). “A Human Capital Model for Mitigating Security Analyst Burnout.” In: *Eleventh Symposium On Usable Privacy and Security ({SOUPS} 2015)*, pp. 347–359.
- Suryn, Witold, Abran, Alain, and April, Alain (2003). *ISO/IEC SQuaRE: The second generation of standards for software product quality*.
- Syed, Zareen, Padia, Ankur, Finin, Tim, Mathews, Lisa, and Joshi, Anupam (2016). “UCO: A Unified Cybersecurity Ontology.” In: *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*.
- Symantec (2022). <https://www.symantec.com/>. Accessed: 2022-04-23.
- Symbiotic Human-Robot Collaborative Assembly: Technologies, Innovations and Competitiveness* (2022). <https://cordis.europa.eu/project/id/637107>. Accessed: 2022-04-23.
- Tang, MingJian, Alazab, Mamoun, and Luo, Yuxiu (2017). “Big Data for Cybersecurity: Vulnerability Disclosure Trends and Dependencies.” In: *IEEE Transactions on Big Data* 5.3, pp. 317–329.
- Tao, Dapeng, Cheng, Jun, Yu, Zhengtao, Yue, Kun, and Wang, Lizhen (2018). “Domain-Weighted Majority Voting for Crowdsourcing.” In: *IEEE Transactions on Neural Networks and Learning Systems* 30.1, pp. 163–174.
- Tavabi, Nazgol, Goyal, Palash, Almukaynizi, Mohammed, Shakarian, Paulo, and Leraman, Kristina (2018). “Darkembed: Exploit Prediction with Neural Language Models.” In: *Thirty-Second AAAI Conference on Artificial Intelligence*.
- TfidfTransformer* (2022). [https://scikit-learn.org/stable/modules/generated/sklearn.feature\\_extraction.text.TfidfTransformer.html](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html). Accessed: 2022-04-23.
- Teixeira, Andre, Sou, Kin Cheong, Sandberg, Henrik, and Johansson, Karl Henrik (2015). “Secure Control Systems: A Quantitative Risk Management Approach.” In: *IEEE Control Systems Magazine* 35.1, pp. 24–45.
- Thomson, S Bruce (2011). “Qualitative research: validity.” In: *JOAAG*.
- Tong, Haonan, Liu, Bin, and Wang, Shihai (2018). “Software Defect Prediction Using Stacked Denoising Autoencoders and Two-Stage Ensemble Learning.” In: *Information and Software Technology* 96, pp. 94–111.

- Torkura, Kennedy Aondona, Cheng, Feng, and Meinel, Christoph (2016). “Aggregating Vulnerability Information for Proactive Cloud Vulnerability Assessment.” In: vol. 4, 2, pp. 387–395.
- Torres, Javier Martínez, Comesaña, Carla Iglesias, and García-Nieto, Paulino J (2019). “Machine Learning Techniques Applied to Cybersecurity.” In: *International Journal of Machine Learning and Cybernetics* 10,10, pp. 2823–2836.
- Trist, Eric (1981). “The Evolution of Socio-Technical Systems.” In: *Occasional paper* 2.1981, p. 1981.
- Trstenjak, Bruno, Mikac, Sasa, and Donko, Dzenana (2014). “KNN with TF-IDF Based Framework for Text Categorization.” In: *Procedia Engineering* 69, pp. 1356–1364.
- Tsoumakas, Grigorios and Katakis, Ioannis (2007). “Multi-Label Classification: An Overview.” In: *International Journal of Data Warehousing and Mining (IJDWM)* 3,3, pp. 1–13.
- Uday, Payuna and Marais, Karen (2015). “Designing Resilient Systems-of-Systems: A Survey of Metrics, Methods, and Challenges.” In: *Systems Engineering* 18,5, pp. 491–510.
- Uslar, Mathias, Rohjans, Sebastian, Neureiter, Christian, Pröstl Andrén, Filip, Velasquez, Jorge, Steinbrink, Cornelius, Efthymiou, Venizelos, Migliavacca, Gianluigi, Horsmanheimo, Seppo, Brunner, Helfried, et al. (2019). “Applying the smart grid architecture model for designing and validating system-of-systems in the power and energy domain: A European perspective.” In: *Energies* 12,2, p. 258.
- Vaiman, Marianna, Bell, Keith, Chen, Yousu, Chowdhury, Badrul, Dobson, Ian, Hines, Paul, Papic, Milorad, Miller, Stephen, and Zhang, Pei (2012). “Risk assessment of cascading outages: Methodologies and challenges.” In: *IEEE Transactions on Power Systems* 27,2, p. 631.
- Välja, Margus, Lagerström, Robert, Franke, Ulrik, and Ericsson, Göran (2018). “A Framework for Automatic IT Architecture Modelling: Applying Truth Discovery.” In.
- Van der Laan, Mark J, Polley, Eric C, and Hubbard, Alan E (2007). “Super learner.” In: *Statistical Applications in Genetics and Molecular Biology* 6,1.
- Vanerio, Juan and Casas, Pedro (2017). “Ensemble-Learning Approaches for Network Security and Anomaly Detection.” In: *Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, pp. 1–6.
- Veksler, Vladislav D, Buchler, Norbou, Hoffman, Blaine E, Cassenti, Daniel N, Sample, Char, and Sugrim, Shridat (2018). “Simulations in Cyber-Security: A Review of Cognitive Modelling of Network Attackers, Defenders, and Users.” In: *Frontiers in Psychology* 9, p. 691.
- Venkata, Rohith Yanambaka, Kamongi, Patrick, and Kavi, Krishna (2018). “An Ontology-Driven Framework for Security and Resiliency in Cyber Physical Systems.” In: *ICSEA 2018*, p. 23.

- VIA4CVE - Vulnerability Information Aggregator for CVEs (2022). <https://github.com/cve-search/VIA4CVE>. Accessed: 2022-04-23.
- Vielberth, Manfred, Böhm, Fabian, Fichtinger, Ines, and Pernul, Günther (2020). “Security operations center: A systematic study and open challenges.” In: *IEEE Access* 8, pp. 227756–227779.
- Vijay, Vaishnavi, Bill, Kuechler, and Stacie, Petter (2015). *Design Science Research in Information Systems*. url: <http://desrist.org/desrist/> (visited on 2015).
- Vukovic, Ognjen, Sou, Kin Cheong, Dan, Gyorgy, and Sandberg, Henrik (2012). “Network-Aware Mitigation of Data Integrity Attacks on Power System State Estimation.” In: *IEEE Journal on Selected Areas in Communications* 30.6, pp. 1108–1118.
- Vulnerability Classifier (2021). <https://github.com/Yuning-J/Vulnerability-Classifier>. Accessed: 2022-04-23.
- Wang, Chaonan, Xing, Liudong, and Levitin, Gregory (2012). “Competing Failure Analysis in Phased-Mission Systems with Functional Dependence in One of Phases.” In: *Reliability Engineering & System Safety* 108, pp. 90–99.
- Wang, Ju An and Guo, Minzhe (2009). “OVM: an Ontology for Vulnerability Management.” In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, pp. 1–4.
- Wang, Ju An, Zhang, Fengwei, and Xia, Min (2008). “Temporal Metrics for Software Vulnerabilities.” In: *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, pp. 1–3.
- Wang, Lihui, Gao, R, Váncza, József, Krüger, J, Wang, Xi Vincent, Makris, S, and Chryssolouris, G (2019). “Symbiotic Human-Robot Collaborative Assembly.” In: *CIRP annals* 68.2, pp. 701–726.
- Wang, Sida and Manning, Christopher D (2012). “Baselines and Bigrams: Simple, Good Sentiment and Topic Classification.” In: *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*. Association for Computational Linguistics, pp. 90–94.
- Waterhouse, Steve, MacKay, David, Robinson, Tony, et al. (1996). “Bayesian Methods for Mixtures of Experts.” In: *Advances in neural information processing systems*, pp. 351–357.
- Wen, Tao, Zhang, Yuqing, Dong, Ying, and Yang, Gang (2015). “A Novel Automatic Severity Vulnerability Assessment Framework.” In: *J. Commun.* 10.5, pp. 320–329.
- Williams, Theodore J (1994). “The Purdue enterprise reference architecture.” In: *Computers in industry* 24.2-3, pp. 141–158.
- Wohlin, Claes (2021). “Case Study Research in Software Engineering: It is a Case, and it is a Study, but is it a Case Study?” In: *Information and Software Technology* 133, p. 106514.

- Wolpert, David H (1992). "Stacked Generalization." In: *Neural networks* 5.2, pp. 241–259.
- Wu, Dazhong, Ren, Anqi, Zhang, Wenhui, Fan, Feifei, Liu, Peng, Fu, Xinwen, and Terpenny, Janis (2018). "Cybersecurity for Digital Manufacturing." In: *Journal of Manufacturing Systems* 48, pp. 3–12.
- Wynn, Jackson (2014). *Threat assessment and remediation analysis (tara)*. Tech. rep. Mitre Corp Bedford Ma Bedford United States.
- Xin, Yang, Kong, Lingshuang, Liu, Zhi, Chen, Yuling, Li, Yanmiao, Zhu, Hongliang, Gao, Mingcheng, Hou, Haixia, and Wang, Chunhua (2018). "Machine Learning and Deep Learning Methods for Cybersecurity." In: *IEEE Access* 6, pp. 35365–35381.
- Xu, Hansong, Yu, Wei, Griffith, David, and Golmie, Nada (2018a). "A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective." In: *IEEE Access* 6, pp. 78238–78259.
- Xu, Maochao, Schweitzer, Kristin M, Bateman, Raymond M, and Xu, Shouhuai (2018b). "Modelling and Predicting Cyber Hacking Breaches." In: *IEEE Transactions on Information Forensics and Security* 13.11, pp. 2856–2871.
- Xu, Shouhuai (2014). "Cybersecurity Dynamics." In: *Proceedings of the 2014 Symposium and Bootcamp on the Science of Security*, pp. 1–2.
- Xu, Shouhuai (2019). "Cybersecurity Dynamics: A Foundation for the Science of Cybersecurity." In: *Proactive and Dynamic Network Defense*. Springer, pp. 1–31.
- Yamamoto, Yasuhiro, Miyamoto, Daisuke, and Nakayama, Masaya (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, pp. 67–73.
- Yazar, Zeki (2002). "A Qualitative Risk Analysis and Management Tool - CRAMM." In: *SANS InfoSec Reading Room White Paper* 11, pp. 12–32.
- Yi, Feng, Jiang, Bo, Wang, Lu, and Wu, Jianjun (2020). "Cybersecurity Named Entity Recognition Using Multi-Modal Ensemble Learning." In: *IEEE Access* 8, pp. 63214–63224.
- Yin, Robert K (2009). "Case Study Research: Design and Methods. Sage Publications." In: *Thousand oaks*.
- Yin, Robert K (2011). *Applications of Case Study Research*. Sage Publications.
- Ying, Zou, Yirong, Wang, and Ning, Wang (2014). "Study of network architecture and IP address allocation of wireless VPN for power grid." In: *2014 Enterprise Systems Conference*. IEEE, pp. 305–309.
- Yuan, Liu, Bai, Yude, Xing, Zhenchang, Chen, Sen, Li, Xiaohong, and Deng, Zhidong (2021). "Predicting Entity Relations across Different Security Databases by Using Graph Attention Network." In: *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, pp. 834–843.

- Zaki, Mohammed J and Hsiao, Ching-Jui (2002). "CHARM: An Efficient Algorithm for Closed Itemset Mining." In: *Proceedings of the 2002 SIAM international conference on data mining*. SIAM, pp. 457–473.
- Zanaty, EA (2012). "Support Vector Machines (SVMs) Versus Multilayer Perception (MLP) in Data Classification." In: *Egyptian Informatics Journal* 13.3, pp. 177–183.
- Zeinali, Mehdi and Thompson, John (2021). "Comprehensive practical evaluation of wired and wireless internet base smart grid communication." In: *IET Smart Grid* 4.5, pp. 522–535.
- Zhang, Su, Caragea, Doina, and Ou, Xinming (2011). "An Empirical Study on Using the National Vulnerability Database to Predict Software Vulnerabilities." In: *International Conference on Database and Expert Systems Applications*. Springer, pp. 217–231.
- Zhao, Guilin and Xing, Liudong (2019). "Competing Failure Analysis Considering Cascading Functional Dependence and Random Failure Propagation Time." In: *Quality and Reliability Engineering International* 35.7, pp. 2327–2342.
- Zhou, Peng, Qi, Zhenyu, Zheng, Suncong, Xu, Jiaming, Bao, Hongyun, and Xu, Bo (2016). "Text Classification Improved by Integrating Bidirectional LSTM with Two-Dimensional Max Pooling." In: *ArXiv Preprint ArXiv:1611.06639*.
- Zhou, Qunzhi, Natarajan, Sreedhar, Simmhan, Yogesh, and Prasanna, Viktor (2012). "Semantic Information Modelling for Emerging Applications in Smart Grid." In: *Information Technology: New Generations (ITNG), 2012 Ninth International Conference on*. IEEE, pp. 775–782.
- Zhu, Wentao and Milanović, Jovica V (2017). "Interdependency modeling of cyber-physical systems using a weighted complex network approach." In: *2017 IEEE Manchester PowerTech*. IEEE, pp. 1–6.
- Zhu, Ziyun and Dumitras, Tudor (2018). "Chainsmith: Automatically Learning the Semantics of Malicious Campaigns by Mining Threat Intelligence Reports." In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, pp. 458–472.
- Zhu, Ziyun and Dumitraş, Tudor (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*. ACM, pp. 767–778.
- Zio, Enrico (2016). "Challenges in the Vulnerability and Risk Analysis of Critical Infrastructures." In: *Reliability Engineering & System Safety* 152, pp. 137–150.
- Zio, Enrico and Sansavini, Giovanni (2013). "Vulnerability of Smart Grids with Variable Generation and Consumption: A System of Systems Perspective." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 43.3, pp. 477–487.



## PUBLICATIONS IN THE DISSERTATION SERIES



# PUBLICATIONS IN THE DISSERTATION SERIES

1. Berg Marklund, Björn (2013) Games in formal educational settings: obstacles for the development and use of learning games, Informatics. Licentiate Dissertation, ISBN 978-91-981474-0-7.
2. Aslam, Tehseen (2013) Analysis of manufacturing supply chains using system dynamics and multi-objective optimization, Informatics. Doctoral Dissertation, ISBN 978-91-981474-1-4.
3. Laxhammar, Rikard (2014) Conformal Anomaly Detection-Detecting Abnormal Trajectories in Surveillance Applications, Informatics. Doctoral Dissertation, ISBN 978-91-981474-2-1.
4. Alklind Taylor, Anna-Sofia (2014) Facilitation matters: a framework for instructor-led serious gaming, Informatics. Doctoral Dissertation, ISBN 978-91-981474-4-5.
5. Holgersson, Jesper (2014) User participation in public e-service development: guidelines for including external users, Informatics. Doctoral Dissertation, ISBN 978-91-981474-5-2.
6. Kaidalova, Julia (2015) Towards a Definition of the role of Enterprise Modeling in the Context of Business and IT Alignment, Informatics. Licentiate Dissertation, ISBN 978-91-981474-6-9.
7. Rexhepi, Hanife (2015) Improving health care information systems – A key to evidence based medicine, Informatics. Licentiate Dissertation, ISBN 978-91-981474-7-6.
8. Berg Marklund, Björn (2015) Unpacking Digital Game-Based Learning: The complexities of developing and using educational games, Informatics. Doctoral Dissertation, ISBN 978-91-981474-8-3.
9. Fornlöf, Veronica (2016) Improved remaining useful life estimations for on-condition parts in aircraft engines, Informatics. Licentiate Dissertation, ISBN 978-91-981474-9-0.
10. Ohlander, Ulrika (2016) Towards Enhanced Tactical Support Systems, Informatics. Licentiate Dissertation, ISBN 978-91-982690-0-0.
11. Siegmund, Florian (2016) Dynamic Resampling for Preference-based Evolutionary Multi-objective Optimization of Stochastic Systems, Informatics. Doctoral Dissertation, ISBN 978-91-982690-1-7.
12. Kolbeinsson, Ari (2016) Managing Interruptions in Manufacturing: Towards a Theoretical Framework for Interruptions in Manufacturing Assembly, Informatics. Licentiate Dissertation, ISBN 978-91-982690-2-4.

13. Sigholm, Johan (2016) Secure Tactical Communications for Inter-Organizational Collaboration: The Role of Emerging Information and Communications Technology, Privacy Issues, and Cyber Threats on the Digital Battlefield, Informatics. Doctoral Dissertation, ISBN 978-91-982690-3-1.
14. Brolin, Anna (2016) An investigation of cognitive aspects affecting human performance in manual assembly, Informatics. Doctoral Dissertation, ISBN 978-91-982690-4-8.
15. Brodin, Martin (2016) Mobile device strategy: A management framework for securing company information assets on mobile devices, Informatics. Licentiate Dissertation, ISBN 978-91-982690-5-5.
16. Ericson, Stefan (2016) Vision-based perception for localization of autonomous agricultural robots, Informatics. Doctoral Dissertation, ISBN 978-91-982690-7-9.
17. Holm, Magnus (2017) Adaptive Decision Support for Shopfloor: Operators using Function Blocks, Informatics. Doctoral Dissertation, ISBN 978-91-982690-8-6.
18. Larsson, Carina (2017) Communicating performance measures: Supporting continuous improvement in manufacturing companies, Informatics. Licentiate Dissertation, ISBN 978-91-982690-9-3.
19. Rexhepi, Hanife (2018) Bridging the Information Gap: Supporting Evidence-Based Medicine and Shared Decision-Making through Information Systems, Informatics. Doctoral Dissertation, ISBN 978-91-984187-1-2.
20. Schmidt, Bernard (2018) Toward Predictive Maintenance in a Cloud Manufacturing Environment: A population-wide approach, Informatics. Doctoral Dissertation, ISBN 978-91-984187-2-9.
21. Linnéusson, Gary (2018) Towards strategic development of maintenance and its effects on production performance: A hybrid simulation-based optimization framework, Informatics. Doctoral Dissertation, ISBN 978-91-984187-3-6.
22. Amouzgar, Kaveh (2018) Metamodel Based Multi-Objective Optimization with Finite-Element Applications, Informatics. Doctoral Dissertation, ISBN 978-91-984187-4-3.
23. Bernedixen, Jacob (2018) Automated Bottleneck Analysis of Production systems, Informatics. Doctoral Dissertation, ISBN 978-91-984187-6-7.
24. Karlsson, Ingemar (2018) An Interactive Decision Support System Using Simulation-Based Optimization and Knowledge Extraction, Informatics. Doctoral Dissertation, ISBN 978-91-984187-5-0.
25. Andersson, Martin (2018) A Bilevel Approach To Parameter Tuning of Optimization Algorithms Using Evolutionary Computing, Informatics. Doctoral Dissertation, ISBN 978-91-984187-7-4.
26. Tavara, Shirin (2018) High-Performance Computing For Support Vector Machines, Informatics. Licentiate Dissertation, ISBN 978-91-984187-8-1.
27. Bevilacqua, Fernando (2018) Game-calibrated and user-tailored remote detection of emotion: A non-intrusive, multifactorial camera-based approach for detecting stress and boredom of players in games, Informatics. Doctoral Dissertation, ISBN 978-91-984187-9-8.

28. Kolbeinsson, Ari (2018) Situating Interruptions in Manufacturing Assembly, Informatics. Doctoral Dissertation, ISBN 978-91-984918-0-7.
29. Goienetxea, Ainhoa (2019) Bringing together Lean, simulation and optimization: Defining a framework to support decision-making in system design and improvement, Informatics. Doctoral Dissertation, ISBN 978-91-984918-1-4.
30. Gudfinnsson, Kristens (2019) Towards Facilitating BI Adoption in Small and Medium Sized Manufacturing Companies, Informatics. Doctoral Dissertation, ISBN 978-91-984918-2-1.
31. Kaidalova, Julia (2019) Integration of Product-IT into Enterprise Architecture: a Method for Participatory Business and IT Alignment, Informatics. Doctoral Dissertation, ISBN 978-91-984918-3-8.
32. Brodin, Martin (2020) Managing information security for mobile devices in small and medium-sized enterprises, Informatics. Doctoral Dissertation, ISBN 978-91-984918-4-5.
33. Bergström, Erik (2020) Supporting Information Security Management: Developing a Method for Information Classification, Informatics. Doctoral Dissertation, ISBN 978-91-984918-5-2.
34. Gustavsson, Patrik (2020) Virtual Reality Platform for Design and Evaluation of the Interaction in Human-Robot Collaborative Tasks in Assembly Manufacturing. Doctoral Dissertation, ISBN 978-91-984918-6-9.
35. Ruiz Zúñiga, Enrique (2020) Facility layout redesign with simulation-based optimization - a holistic methodology including process, flow, and logistics constraints in manufacturing. Doctoral Dissertation, ISBN 978-91-984918-9-0.
36. Ventocilla, Elio (2021) Visualizing Cluster Patterns at Scale. A Model and a Library. Doctoral Dissertation, ISBN 978-91-984919-0-6.
37. Danielsson, Oscar (2020) Augmented reality smart glasses as assembly operator support: Towards a framework for enabling industrial integration. Licentiate Dissertation, ISBN 978-91-984919-1-3
38. Morshedzadeh, Iman (2021) Manging virtual factory artifacts in extended product lifecycle management systems. Doctoral Dissertation, ISBN 978-91-984919-2-0.
39. Ståhl, Niclas (2021) Integration of expert knowledge in deep learning models for increased model performance. Doctoral Dissertation, ISBN 978-91-984919-3-7.
40. Lidberg, Simon (2021) Evaluating fast and efficient modeling methods for simulation-based optimization. Licentiate Dissertation, ISBN 978-91-984919-4-4.
41. Senavirathne, Navoda (2021) Towards privacy preserving micro-data analysis – A machine learning based perspective under prevailing privacy regulations. Doctoral Dissertation, ISBN 978-91-984919-5-1.
42. Lennerholt, Christian (2022) Facilitating the implementation and use of self service business intelligence. Doctoral Dissertation, ISBN 978-91-984919-6-8.
43. Liu, Yu (2022) Integrating life cycle assessment into simulation-based decision support. Licentiate Dissertation, ISBN 978-91-984919-7-5.

44. Tavara, Shirin (2022) Distributed and federated learning of support vector machines and applications. Doctoral Dissertation, ISBN 978-91-984919-8-2.
45. Kävrestad, Joakim (2022) Context-Based Micro-Training – the development of a method for cyber security training of end-users. Doctoral Dissertation, ISBN 978-91-984919-9-9.