

---

# Contents

<b>List of Abbreviations</b>	<b>xiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Security Community Knowledge . . . . .	2
1.2 Challenges . . . . .	4
1.3 Research Questions and Methodology . . . . .	6
1.4 Contributions . . . . .	8
1.5 Structure . . . . .	8
<b>2 Background</b>	<b>11</b>
2.1 Vulnerabilities, Patches, and Exploits . . . . .	11
2.2 Data, Information, and Knowledge . . . . .	12
2.3 Sources of Security Knowledge . . . . .	13
2.3.1 Vulnerability Databases . . . . .	13
2.3.2 Question and Answer Websites . . . . .	15
2.3.3 GitHub . . . . .	16
2.4 Artificial Intelligence for Text Classification . . . . .	16
2.5 Code Clone Detection . . . . .	17
2.5.1 Code Clone Types . . . . .	18
2.6 Performance Metrics . . . . .	20
2.7 Threats to Validity . . . . .	22
<b>3 Heuristic Security Checker</b>	<b>23</b>
3.1 Software Code Artifacts . . . . .	24
3.2 Integration of Approach into Software Development . . . . .	25
3.3 Programming Language Selection as Research Base . . . . .	27

3.4	General Approach . . . . .	27
3.4.1	Study: Community Knowledge about Security . . . . .	29
3.5	Security Heuristics . . . . .	31
3.6	Code Clone Detection . . . . .	32
3.7	Related Work . . . . .	33
<b>4</b>	<b>Security Knowledge Acquisition from Coding Communities</b>	<b>35</b>
4.1	General Approach . . . . .	36
4.2	Identification of Security-Related Information . . . . .	37
4.2.1	Keyword-Based Search . . . . .	39
4.2.2	Data about Security . . . . .	41
4.2.3	Combining Artificial Intelligence and Natural Language Processing . . . . .	44
4.2.4	Filter for Programming Language . . . . .	49
4.3	Classification of Security-Related Information . . . . .	50
4.4	Postprocessing of Data: CWE Assignment, Relationship between Posts and Verification . . . . .	52
4.5	Related Work . . . . .	53
<b>5</b>	<b>Identification of Vulnerabilities in Software Libraries</b>	<b>55</b>
5.1	Manual Approach to Identify Insecure Software Libraries . . . . .	56
5.2	Sources and Challenges for Identifying Software Libraries . . . . .	57
5.2.1	Filenames . . . . .	57
5.2.2	JAR-Manifest . . . . .	58
5.2.3	Build Systems . . . . .	61
5.2.4	File-Hashes . . . . .	62
5.2.5	JavaScript File . . . . .	62
5.2.6	Node Package Manager . . . . .	64
5.2.7	Relevance of Sources . . . . .	65
5.3	Metadata Library Checker . . . . .	66
5.3.1	General Metadata Library Checker Approach . . . . .	66
5.3.2	Search on the NVD . . . . .	67
5.3.3	Scope and Limitation for other Programming Languages . . . . .	71
5.4	File Hash Library Checker . . . . .	72

5.4.1	General Hash-based Library Checker Approach . . . . .	73
5.4.2	Hash Database Creation Process - Java . . . . .	74
5.4.3	Hash Database Creation Process - JavaScript . . . . .	79
5.4.4	Library Hash Checker for Projects . . . . .	81
5.5	Related Work . . . . .	83
<b>6</b>	<b>Identification of Vulnerabilities in Source Code</b>	<b>87</b>
6.1	Challenges . . . . .	88
6.2	General Approach . . . . .	89
6.3	Java vs. JavaScript Source Code . . . . .	91
6.4	Code Clone Detection Approach and its Limitations . . . . .	92
6.4.1	Adaption of the SourcererCC . . . . .	94
6.5	Heuristics . . . . .	96
6.5.1	Jaccard-Index . . . . .	97
6.5.2	Salton's Cosine . . . . .	98
6.5.3	Patch-Comparison . . . . .	98
6.5.4	Weighting of Tokens . . . . .	99
6.5.5	Borderline . . . . .	103
6.6	Related Work . . . . .	104
<b>7</b>	<b>Prototypical Implementation</b>	<b>107</b>
7.1	Security Checker Eclipse Plugins . . . . .	107
7.1.1	Eclipse Library Checker . . . . .	107
7.1.2	Eclipse Code Clone Detection . . . . .	109
7.2	JIRA Security Checker . . . . .	110
<b>8</b>	<b>Technical Validation</b>	<b>113</b>
8.1	Security Knowledge Base Creation . . . . .	113
8.1.1	Identification of Security-Related Code Fragments . . . . .	113
8.1.2	Classification of Security-Related Code Fragments . . . . .	117
8.1.3	Threats to Validity . . . . .	120
8.2	Library Checker . . . . .	121
8.2.1	Java . . . . .	121

8.2.2	JavaScript . . . . .	123
8.2.3	Threats to Validity . . . . .	124
8.3	Security Code Clone Detection . . . . .	125
8.3.1	Java . . . . .	125
8.3.2	JavaScript . . . . .	132
8.3.3	Threats to Validity . . . . .	133
<b>9</b>	<b>Case Study: Applying Security Checks Based on Community Knowledge to Software Projects</b>	<b>135</b>
9.1	Security Knowledge Base Validation . . . . .	135
9.1.1	Methodology . . . . .	136
9.1.2	Results and Discussion . . . . .	136
9.1.3	Threats to Validity . . . . .	138
9.2	Applying Jira Security Checker to Real Projects . . . . .	138
9.2.1	Projects . . . . .	139
9.2.2	Methodology and Change Requests . . . . .	139
9.2.3	Vulnerability Results . . . . .	143
9.2.4	Threats to Validity . . . . .	145
9.3	Applying Eclipse Security Checkers within an Experiment . . . . .	146
9.3.1	Methodology . . . . .	146
9.3.2	Results and Discussion . . . . .	147
9.3.3	Threats to Validity . . . . .	149
<b>10</b>	<b>Conclusion</b>	<b>151</b>
10.1	Addressed Challenges . . . . .	151
10.2	Answers to Research Questions . . . . .	153
10.3	Limitations of Heuristic Security Knowledge Checks . . . . .	154
10.4	Future Work . . . . .	155
<b>A</b>	<b>Appendix</b>	<b>157</b>
A.1	Preliminary Publications . . . . .	157
A.2	Dissertation Context: Research Project SecVolution . . . . .	160
A.3	Java Libraries used for Manifest-File Analysis . . . . .	164

A.4	Metadata Library Checker Configurations . . . . .	164
A.5	Calculation Heuristics . . . . .	165
A.6	Prototypical Implementation: Security Knowledge Base Creation . . . . .	166
A.6.1	Security Knowledge Base Database Structure . . . . .	167
A.7	Heuristic Performances and Iteration Weights . . . . .	168
A.8	Experiment with Security Experts to Identify Security-Related Stack Over- flow Posts . . . . .	169
A.8.1	Demographic Survey and Search Task . . . . .	169
A.8.2	Classification Task . . . . .	173
A.8.3	Classification Tool . . . . .	176
A.8.4	Stack Overflow Posts Used for Classification . . . . .	176
A.9	Survey for the Experiment Case Study . . . . .	177
A.9.1	Code Fragments for the Security Assessment . . . . .	186
	<b>Bibliography</b>	<b>189</b>
	<b>List of Figures</b>	<b>201</b>
	<b>List of Tables</b>	<b>203</b>
	<b>Curriculum Vitae</b>	<b>205</b>