INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Low-Code Security for Industrial Applications

Miguel da Ponte Lourenço

Master's in Telecommunications and Computer Engineering

Supervisor:
Doctor Maria Cabral Diogo Pinto Albuquerque, Assistant Professor,
ISCTE - Instituto Universitário de Lisboa, Portugal

Co-Supervisor:
Doctor Tiago José Espinha de Mendonça Gasiba, Senior Key Expert,
Siemens AG, Munich, Germany

October, 2023

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

Low-Code Security for Industrial Applications

Miguel da Ponte Lourenço

Master's in Telecommunications and Computer Engineering

Supervisor:
Doctor Maria Cabral Diogo Pinto Albuquerque, Assistant Professor,
ISCTE - Instituto Universitário de Lisboa, Portugal

Co-Supervisor:
Doctor Tiago José Espinha de Mendonça Gasiba, Senior Key Expert,
Siemens AG, Munich, Germany

October, 2023

Acknowledgments

These past months have been really challenging with new experiences that I could never imagine going through them. The research and development described in this master's thesis were much more than the information presented in it: it is also the new challenges that I faced in these last months that resulted in this thesis. Throughout the writing of this thesis, there have been ups and downs, but I always had support every step of the way. I am grateful for the people who have helped me through this arduous but gratifying process.

In March of 2023, I went to Munich to work at Siemens AG and develop the research presented in this thesis. I am thankful to my supervisor Maria Pinto Albuquerque for her everlasting help and meticulous monitoring that facilitated the development and writing of this thesis. My time at Siemens was productive and working with my co-supervisor was an incredible experience. For this, I am utterly thankful to my co-supervisor Tiago Gasiba. Not only was he exceptionally helpful and always ready to aid me when I needed it, but he also pushed me to surpass my limits and was always ready to teach me different topics. Without their assistance, dedication and involvement in every step of this work, this dissertation would not have been achieved. I would like to thank you both for all the support during these last months. Without you, this experience would not have been the same and for that I am grateful.

Writing this thesis needed more than just academic assistance, for which I am grateful to many people. To my family for the constant support and being present even when at a distance in these past months. To my friends for all the fun, laughs and kind words that motivated me to write this thesis.

Resumo

As *Low-Code Development Platforms* (LCDPs) estão a ganhar cada vez mais força, mesmo no contexto industrial, como forma de tornar o desenvolvimento de software mais rápido, barato e fácil. Com as suas características visuais, como as interfaces gráficas de fácil utilização e o recurso ao *drag-and-drop*, qualquer pessoa, desde especialistas em programação a pessoas com pouca ou nenhuma experiência em desenvolvimento, pode utilizá-las para desenvolver e implementar aplicações. No entanto, pouco se sabe sobre as vulnerabilidades resultantes deste novo modelo de desenvolvimento de software. Apesar de qualquer pessoa poder desenvolver software com LCDPs, as pessoas com menos conhecimentos de cibersegurança podem, involuntariamente, adicionar vulnerabilidades às suas aplicações. Esta tese tem como objetivo compreender as vulnerabilidades das aplicações desenvolvidas e implementadas nestas plataformas, abordando o problema das vulnerabilidades nas LCDPs através do desenvolvimento de um artefacto. As vulnerabilidades podem ser consideradas a partir de três perspectivas: plataforma, programador e *plug-ins*. O artefacto apresenta um top três de vulnerabilidades para cada perspetiva, baseado numa revisão da literatura, pesquisa em bases de dados e entrevistas a especialistas. Além disso, são fornecidas directrizes sobre como desenvolver aplicações de forma segura através destas plataformas, com base na informação sistematizada sobre as vulnerabilidades. Os resultados mostram que o artefacto desenvolvido é um bom método para compreender o problema definido e foi aceite na indústria para a qual foi criado. Este trabalho contribui para a compreensão da segurança das aplicações desenvolvidas com LCDPs e sensibiliza os profissionais do sector, sistematizando informação sobre cibersegurança em LCDPs.

Palavras-Chave: *Low-Code*; Desenvolvimento de Software; Cibersegurança; Indústria; *Low-Code Development Platforms*; Vulnerabilidades.

Abstract

Low-Code Development Platforms (LCDPs) are gaining more and more traction, even in the industrial context, as a means for making software development faster, cheaper and easier. With its visual features, such as user-friendly graphical interfaces and the use of drag-and-drop, anyone from programming experts to someone with less or no experience in development can use them to develop and deploy applications. However, little is known about the vulnerabilities resulting from this new software development model. Although anyone can develop software with LCDPs, people with less cybersecurity knowledge can unwittingly add vulnerabilities to their applications. This thesis aims to understand the vulnerabilities of applications developed and deployed on these platforms, addressing the problem of vulnerabilities in LCDPs by developing an artefact. These vulnerabilities can be considered from three perspectives: platform, developer, and plugins. This artefact presents a top three vulnerabilities for each perspective, based on a literature review, database research and interviews with experts. Also, guidelines are provided on how to develop applications securely using these platforms, based on the systematised information on vulnerabilities. The results show that the artifact developed is a good method for understanding the problem defined and has been accepted in the industry for which it was created. This work contributes to understanding the security of applications developed with LCDPs and raises awareness among professionals in the sector by systematising information on cybersecurity in LCDPs.

Keywords: Low-Code; Software Development; Cybersecurity; Industry; Low-Code Development Platforms; Vulnerabilities.

Contents

CHAPTER 1

Introduction

Low-Code Development Platforms (LCDPs) are an evolution from the Rapid Application Development tools (RAD tools) [**1**], used in the 1990s and early 2000s. The origins of LCDPs may be traced back to fourth-generation programming languages and fast application development tools in the 1990s and early 2000s. LCDPs, like their predecessors [**2**,**3**], are founded on the ideas of model-driven design, automated code generation, and visual programming. LCDPs were first defined by Forrester [**4**,**5**], in 2014, as platforms that "enable rapid delivery of business applications with a minimum of hand-coding and minimal upfront investment in setup and deployment." Subsequently, Gartner [**6**] also described LCDPs as being "platforms that abstract away from code and offer an integrated set of tools to accelerate app delivery". Despite this, Low-Code platforms should not be confused with No-Code platforms [**7**]. While Low-Code platforms allow developers to implement manual code, No-Code platforms depend entirely on the visual tool that they offer. Figure 1.1 shows a window of the Low-Code Development Platform Mendix [**8**].

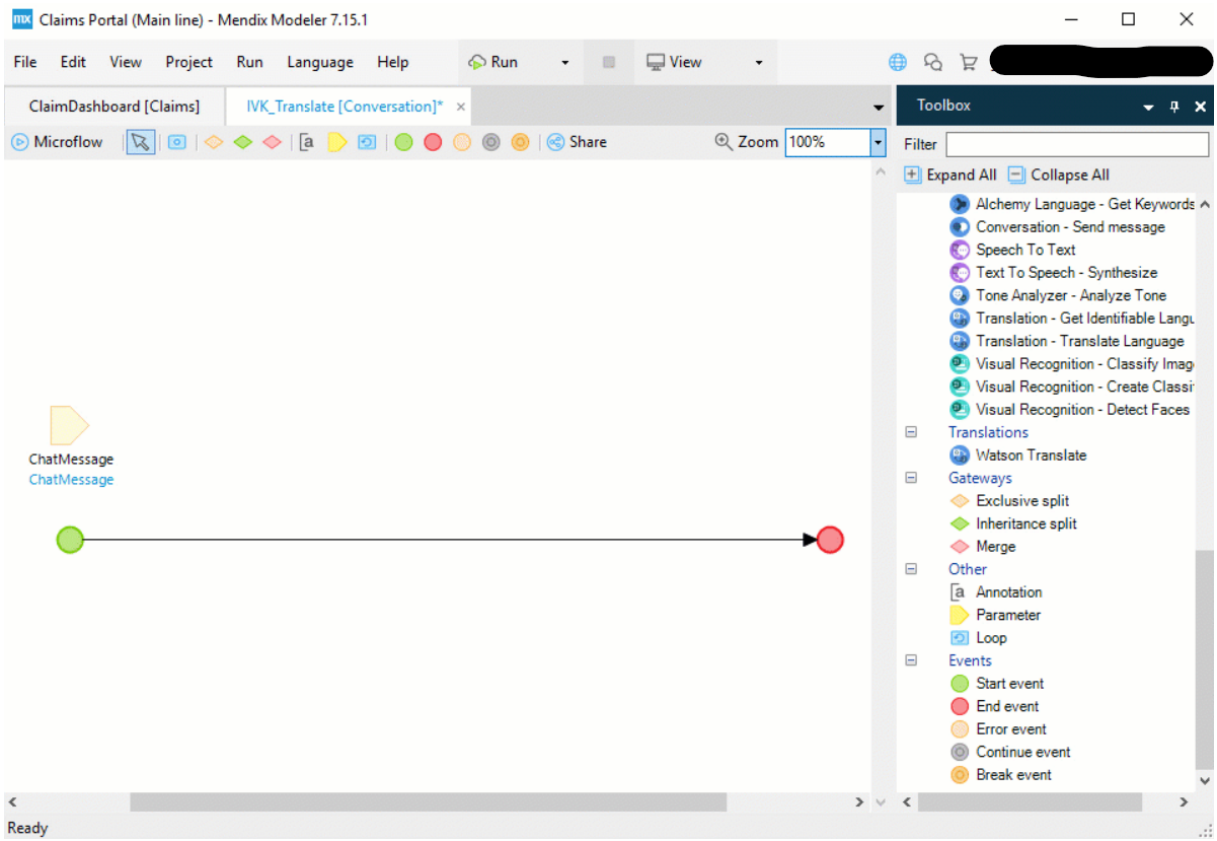

FIGURE 1.1. Example of interface from Mendix

In early 2021, Gartner forecasted [**9**] that the global Low-Code Development Technologies market would be worth $13.8 billion in 2021, a 22.6 percent rise from 2020. According to Gartner, despite continuous cost-cutting measures, the spike in remote development during the COVID-19 epidemic would continue to drive Low-Code adoption, and Low-Code as a broad social and technological movement was predicted to expand greatly in the next years. In accordance with Gartner, the Low-Code Development Technologies market can be defined as a set of technologies, such as Low-Code Development Platforms (LCDPs), Intelligent Business Process Management Suites, Multiexperience Development Platforms (MDXP), Robotic Process Automation (RPA), Citizen Automation and Development Platform (CADP), and Other Low-Code Development (LCD) Technologies. LCDPs, for example, were predicted to be the largest component of the Low-Code Development Technology industry through 2022, rising over 30% from 2015. At the end of 2022, Gartner made another forecast [**10**] on the global market for the Low-Code Development Technologies, expected to reach $26.9 billion in 2023, representing a 19.6% rise from the forecast for 2022. In this study, Gartner's prediction for 2021 was surpassed: instead of an increase to $13.8 billion in revenue on the market, it reached $18.5 billion dollars. Gartner stated that a rising number of enterprise-wide hyperautomation [**11**] and composable business activities [**12**], as well as an increase in business technologists, would be the primary drivers pushing the use of Low-Code solutions through 2026. LCDPs were expected to be the most important component of the Low-Code development technology industry, increasing by 25% to approximately $10 billion by 2023. Figure 1.2 shows the predicted and observed revenues from these two studies by Gartner.

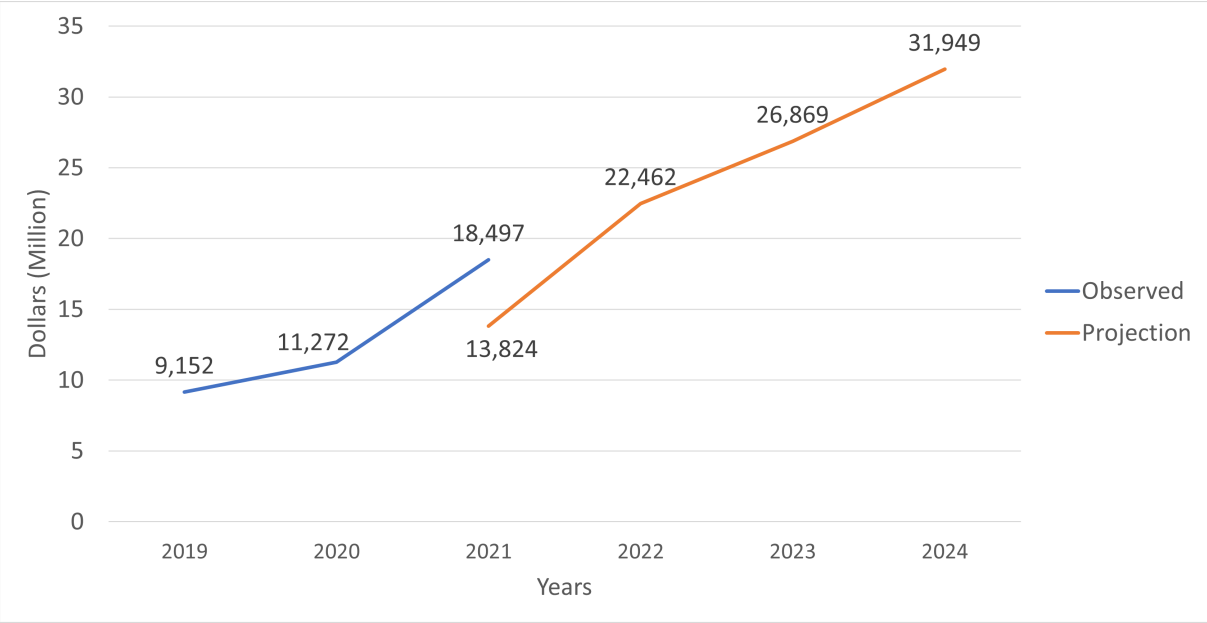

FIGURE 1.2. Forecasts for the revenue on the global Low-Code Development Technologies Market, based on two studies from Gartner [**9**, **10**]

According to Mendix [**13**], there are advantages to using LCDPs over code-based development, such as 1) collaboration, 2) effortless legacy modernization, 3) flexibility and scalability, 4) improved customer experiences, and 5) speed. Collaboration through the use of a LCDP is achieved by means of a visual language between professional developers and other teams in a collaborative atmosphere. This makes designing, deploying and modifying the software possible by aligning different parties involved in the project with the use of a visual interface and drag-and-drop features. According to Mendix [**13**], the modernization of legacy systems is crucial for digital transformation programmes. Even though starting such programs is a big endeavour, Low-Code platforms alleviate the difficulties associated with dealing with these systems. For this purpose, LCDPs allow the creation of apps that integrate with existing software landscapes, extend the capabilities of the legacy systems, and can incrementally replace the systems.

Also, since Low-Code platforms are mostly cloud-based, it allows the deployment of new apps and modification of old software as needed. Users of the platform may onboard fast and simply, system administrators may manage the ecosystem and implement role-based access, and professional developers may customise programs in an Integrated Development Environment (IDE), offering Low-Code apps a competitive advantage over No-Code apps. As stated by Mendix [**13**], competition is fierce and differentiating oneself is becoming increasingly difficult. Customer-centricity is key for survival and Low-Code may assist in providing consistent digital customer experiences. Low-Code Development Platforms allows the user to create customer-facing mobile applications, web portal, Internet-of-Thing (IoT) enabled apps, and more in a single platform. Speed is the core of any LCDP, with capabilities like as visual modelling, pre-built components, automation, one-click deployment, support bots, and built-in monitoring, states Mendix [**13**]. As shown in a statistical article by G2 [**14**], Low-Code development may save development time by up to 90%.

Despite the LCDPs offering many advantages compared to code-based development, the cybersecurity topic has been coming up lately. Forrester analysts predicted that, as more organisations began to adopt Low-Code software development technologies through citizen developers, a big corporate security breach would happen in 2023 [**15**, **16**]. Microsoft's Detection and Response Team (DART) report [**17**] has shown a cybersecurity failure that happened to a large multinational company. In this report it is stated that the attackers used a tool from their LCDP to search for passwords and intellectual property from the employees, performing a *Password Spraying Attack* to the platform and gaining access to sensitive information. A *Password Spraying Attack* [**18**] is a type of brute force attack on which the attackers force the default password on a list of usernames on the application, avoiding account lockouts that would normally occur when trying different passwords for the same account. As stated by OWASP, this attack can be found commonly where the application or administrators set default passwords for the new users, which might have facilitated access to the hackers. The attackers were able to sustain

continuous Office 365 access for 240 days, making use of a *Living-of-the-Land* attack. In accordance with Kaspersky, a *Living-of-the-Land* [**19**] attack defines a cyberattack in which intruders exploit legitimate software and system features to execute harmful acts on the system. This term means surviving on what one can forage, hunt or grow in nature, but transposing to the cybersecurity field, means the trespassers forage on the target systems for tools and use them to achieve their goals. During this period, the attackers were able to find sensitive material and exfiltrate data — all without installing malware or gaining access to the corporate network. In the end, the company saw almost 40 million confidential records and emails being exposed to the Internet.

The motivation for this work came from the necessity of the industry to understand the vulnerabilities of the LCDPs and how to securely develop software using them. In order to write this thesis, I was given the opportunity to be in Munich working at Siemens AG, between the months of March and July 2023. During this time, a conference paper [**20**] was written and accepted, which analyses the most common vulnerabilities in LCDPs and describes the core of the work done in this thesis. Also, interviews with industry experts were conducted in order to obtain information from their field experience. Accordingly, the research questions for the present work are: RQ1: What are the possible threats and vulnerabilities for Low-Code applications? RQ2: How to model risk for Low-Code software? RQ3: What are important aspects of the security life-cycle for Low-Code applications? In this work, we want to address these issues, increase our knowledge of LCDP vulnerabilities and understand how to counter these weaknesses. Therefore, our work aims to generate an artifact - a list of relevant vulnerabilities that can affect applications developed and deployed using LCDPs - and a set of guidelines to help develop secure software through LCDPs. This study approaches the issues by 1) conducting a Lightweight Systematic Literature Review relevant to the topic, 2) performing relevant database searches for known vulnerabilities, and 3) conducting interviews with cybersecurity experts from the industry. Figure A.1 describes the duration of all the activities surrounding this work and when they took place.

The present thesis is structured in the following way. Chapter 2 overviews previous work related to the present study. Chapter 3 provides details on the research method, describes our approach to the problem, and also provides a description of the experiment setup. Results are presented in Chapter 4 and are discussed in detail in Chapter 5. This work is concluded in Chapter 6, which provides a quick overview of our main results and details for further work. In the end, Appendices A, B, and C show information regarding the research of this thesis.

CHAPTER 2

State-of-the-Art

This section characterizes the series of standards that influenced this work, discuss relevant repositories on the topic and provides a description of what LCDPs are and how they operate.

The IEC 62443 series [**21**], is designed to secure Industrial Automation and Control Systems (IACS) throughout their life-cycle, including nine standards, technical reports, and technical specifications. Cyber-attacks on IACS can have economic consequences, environmental or public health risks, and can threaten public health and lives. IEC 62443 addresses not only the technology of a control system but also work processes, countermeasures, and employees. It takes a holistic approach, recognizing that not all risks are technology-based. Staff responsible for an IACS must have the required training, knowledge, and skills to ensure security. The IEC 62443 series is organized into four parts: General, System, Components, and Conformity Assessment. Part 1 covers common topics to the entire series of standards, such as terminologies and concepts. Part 2 centres on the strategies and activities related to IACS security. Part 3 is concerned with systems requirements, such as security requirements and risk assessment. Part 4 offers specifications for IACS products, including secure product development life-cycle requirements and technical security. This series of standards is relevant to this work because it encompasses security at an industrial level for the systems, the solution and the development process for the product, which is what the topic of this work is about.

The ISO/IEC 27000 family of standards [**22**], also known as the ISMS family of standards or ISO27K, covers a broad range of information security standards published by both the International Organisation for Standardisation and the International Electrotechnical Commission. ISO 27000 series recommends best practices for managing information risks by implementing security controls within the framework of an overall Information Security Management System (ISMS). The ISO 27000 series is not specific to any industry, making it applicable to any business, regardless of size and industry. Organizations should tailor their information security controls to treat risks as they deem appropriate. Security guidance and suggestions should be relied upon when appropriate. The ISMS concept incorporates continuous feedback and improvements to respond to changes in threats or vulnerabilities that occur as a result of incidents. Compliance with the ISO 27000 series is the first step toward an information security program that will properly protect an organization. Implementing the ISO 27000 series standard has several benefits, including safeguarding mission-critical data, employee and customer information, instilling greater confidence in operations, enhancing public perception, and providing methodologies for

more effective information security management. It is important to note that the ISO 27000 set of standards is transparent but can be revised as new technology and challenges emerge. This family of standards is relevant for the present work because it was designed to help companies manage cyberattack risks and data security threats.

Gartner [**6**] is a consultant company that offers customers insights, advice and solutions in various areas of IT, such as software development, cloud computing, and data science. This company is important for this work since it provides information regarding the relevant LCDP providers in the market. For this effect, the Magic Quadrant repository for *Enterprise Low-Code Application Platforms* will be used. The Magic Quadrant [**23**] is a repository that provides information on the relevant players in the market, according to Gartner. As reported by the latest version of *Magic Quadrant for Enterprise Low-Code Application Platforms* [**24**], dated the end of 2022, the market leaders are Mendix [**8**], OutSystems [**25**], Microsoft Power Apps [**26**], ServiceNow [**27**], and Salesforce [**28**].

The MITRE Corporation [**29**], commonly known as MITRE, is an American not-for-profit organization that manages federally funded research and development centres (FFRDCs) supporting the United States government agencies in homeland security and cybersecurity fields, among others. In the cybersecurity area, MITRE's projects that are relevant to this work are the Common Vulnerabilities and Exposures and the Common Weakness Enumeration.

The Common Vulnerabilities and Exposures standard (CVE) [**30**], is a list of information security vulnerabilities and exposures, aiming to provide common names for publicly known problems, making data sharing easier across different vulnerability capabilities. In this database, it is possible to search for a development platform or software and be presented with a list of security vulnerabilities regarding it. Amongst these vulnerabilities, we get access to valuable information, for example, the vulnerability's details, the exploit prediction, a score that reflects the severity of the vulnerability, and a Common Weakness Enumeration identification.

The Common Weakness Enumeration standard (CWE) [**31**] is a community-developed list of common software and hardware weaknesses with security implications. According to CWE, a "weakness" is a condition in a component that could introduce vulnerabilities, and the CWE List and classification taxonomy help identify and describe these weaknesses. Each entry on this list has a specific identification number to define the weakness, referred as CWE-ID. Vulnerabilities such as "Cross-Site Scripting" (CWE-79), "SQL Injection" (CWE-89), and "Improper Input Validation" (CWE-20) are presented in this list. As of 2023, the CWE standard identifies over 1000 software vulnerability types. In the present work, the CWE will be used as a standardized means to identify and classify security weaknesses. The usage of CWE for LCDPs is novel because CWE typically applies to software code and LCDPs are Low-Code code. Nevertheless, we base our vulnerability description on CWE as this is a standard provided by MITRE and covers software weaknesses.

The OWASP Top 10 [**32**] is a widely recognized document for web application security, outlining the most significant security threats to web applications. This standard is maintained by the OWASP Foundation [**33**], which enhances software security through its community-led projects. OWASP Top 10 from 2021 stages a diagram with the changes that took place from the last iteration in 2017, explaining the changes and explaining the vulnerability briefly. The vulnerabilities presented in this document are mapped to CWE, also helping in identifying vulnerabilities. This standard is also used in this work because, since the final product of LCDPs is typically web applications, it is natural to use a web vulnerability standard to classify vulnerabilities of web applications.

The goal of this work is to answer the research question that has been raised by the industry. In order to do this, we need to define what are Low-Code Development Platforms and comprehend their characteristics and functionalities. Low-Code Development Platforms are software development tools that offer a visual language, drag-and-drop interface, pre-built components, and templates for developing applications. As such, LCDPs enable either experienced or inexperienced people to develop software, depending on their preferences and requirements. For the inexperienced crowd that develops apps through these platforms, the term Citizen Developers was coined, as described by Gartner [**34**].

The common features present in LCDPs are visual IDE, automated code generation, model-driven development approach, integrations, multi-user features, multi-platform deployment, life-cycle manager and reusable components. The visual IDE facilitates the development through the use of drag-and-drop, making it interactable for the user. Model-Driven Development (MDD) [**35**] is a methodology for software development that enables developers to write and implement software faster, effectively, and at a minimum cost. The LCDPs implement this methodology by abstracting from the technical aspects of the software development, presenting them through visual components. Multi-user features promote the collaboration of multiple users in the same app at the same time and also feature version control between users. Multi-platform deployment allows users to deploy their apps across web, mobile and cloud environments. Life-cycle manager provides the users with tools to build, debug, deploy and maintain apps in multiple environments. Reusable components give the users libraries of pre-built components that can be used to build apps. These platforms, despite having common features, can be focused on different aspects of software development so, it is important to choose the platform in accordance with the type of software to be developed.

As the name implies, the LCDPs do not require big amounts of code; rather small code implementations are possible. This code implementation is made available in case the user pretends to do customization according to their knowledge. It is important to note that Low-Code and No-Code are two separate things: while No-Code does not provide the means to implement custom code from the user, Low-Code makes it happen.

CHAPTER 3

Methodology & Approach

The present chapter describes the methodology followed in this work, separating the research process into two focuses: the research method implemented and the approach taken to conduct the research.

3.1. Research Method

For the present work, we took inspiration from the Design Science Research method by Hevner et al.. Design Science Research (DSR) [**36**] is a research approach that focuses on the creation and validation of prescriptive knowledge in the field of information science. This methodology focuses on creating and enhancing designed artifacts with the explicit aim of enhancing their functional performance. In 2004, Hevner et al. stated that the DSR aims to gain knowledge and understanding of a problem domain by creating and implementing a designed artifact. The authors provide guidelines for design science research in Information Systems, requiring the creation of innovative artifacts for specific problem domains, rigorous construction and evaluation, and effective presentation of results to both technology-oriented and management-oriented audiences.

Thereby, the authors presented seven guidelines for a DSR: design as an artifact, problem relevance, design evaluation, research contributions, research rigor, design as a search process, and communication of research. Design as an artifact is the fact that the DSR necessitates the creation of a viable artifact, such as a construct, model, method, or instantiation. Problem relevance, as the name implies, shows that DSR aims to create technology-based solutions to significant and pertinent business issues. Design evaluation demands that the effectiveness and utility of a design artifact must be thoroughly assessed through effective evaluation methods. Research contributions refer to the fact that DSR should offer clear and verifiable contributions in design artifact, design foundations, andor design methodologies. Research rigor shows that for the production and assessment of design artefacts, DSR applies rigorous techniques. Design as a search process iterates that pursuing an effective artefact demands making use of existing resources to attain desired results while following problem-solving laws. Communication of research DSR must be successfully communicated to both technical and managerial audiences.

Based on these guidelines provided by Hevner et al., four relevant guidelines were adopted for this work: 1) design as an artifact, 2) problem relevance, 3) contributions, and 4) rigor. The designed artifact in this work consists of a table of the top three vulnerabilities and guidelines for more secure development of software with LCDPs. Furthermore, this work's problem is relevant for the industry since cybersecurity is essential

in developing products and services. This thesis aims to shed light on this problem in order to understand it better. Also, it achieves rigor in the research by using diversified sources of information. In particular, this work makes use of existing information in databases and blog posts and validates it through cybersecurity experts' opinions and experience. Regarding the contributions guideline, this work aims to contribute to a better understanding of vulnerabilities in Low-Code Development Platforms. The present work and the conference paper [**20**] produced contribute to academia by deepening the existing knowledge on this subject for future generations. It also contributes to Siemens by improving cybersecurity awareness and the software development process.

The topic that this work addresses can be named a wicked problem. A wicked problem [**37**] is a problem with many interdependent factors making it seem impossible to have a solution. This can be claimed because the topic is unstable thanks to the fact that technology is constantly evolving. The knowledge in this field can become obsolete overnight, and that gives space for new information. Additionally, the requirements keep on changing because the work and results keep progressing, which drives the need for new and better knowledge. Furthermore, this problem is dependent on landscape, culture, and environment because it is a situated research carried out at the company. Even though this study was developed at a specific company, it can aid other companies with similar problems. Moreover, the approach carried out in this work was done to try to gather information, either theoretical or practical, from different sources. Therefore, it can be affirmed that the results present in this work are not universally valid.

To better understand the vulnerabilities of each of the perspectives, we designed an approach that would fit our research. This approach not only covers theoretical research but also a more practical one.

3.2. Approach

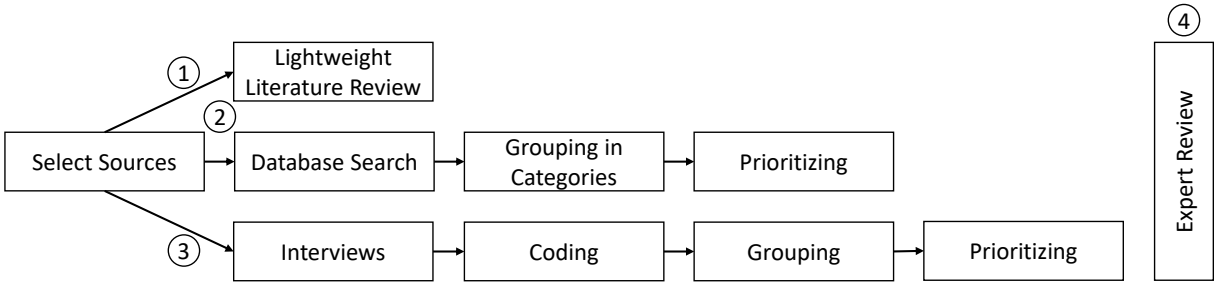

FIGURE 3.1. Research activities

To reach this thesis' goal, an approach was adopted as shown in Figure 3.1. This approach was conducted during the time presented in Figure A.1 and consists of four paths: 1) Lightweight Literature Review, 2) Database Search, 3) Interview, and 4) Expert Review. A detailed description of each path will be presented afterwards. We believe that this approach is appropriate for the topic at hand because it combines information from the literature and from the practical experience of industry experts.

Table 3.1. Keywords and Inclusion and Exclusion Criteria used in LWLR

Keywords	Inclusion Criteria	Exclusion Criteria
"Low-Code"	Documents are single works (articles, conference papers or book chapters)	Works published before 2020
"Low-Code Development"	Works discuss Low-Code	Works not written in English
"Low-Code Platform"	Works are available in digital form	Works that are secondary or tertiary studies
"Low-Code Development Platform"		
"Security in Low-Code"		
"Security in Low-Code Development"		

3.2.1. Lightweight Literature Review

For the first path of the approach, a Literature review was conducted inspired by the Systematic Literature Review (SLR) by Kitchenham et al. [**38**]. In this case, this work's goal is to better understand vulnerabilities related to LCDPs and the software created by them. Because of this, we believe it to be an appropriate Literature Review method to understand the topic addressed in this work. Because of this work's goal, we believe that following this Literature Review method is one way of obtaining information. Therefore, a Lightweight Literature Review was conducted. The Lightweight Literature Review (LWLR) differs from the original method in terms of the amount of time taken, the number of articles used to understand the state-of-the-art on the topic, and snowball research was not used. To engage in this method, a review protocol was created. A total of five databases were used in our search, with the use of a set of keywords and inclusion and exclusion criteria.

The search engines accessed during the research were IEEE [**39**], ACM [**40**], Springer [**41**], Research Gate [**42**], and Google Scholar [**43**]. In order to search for works on the search engines, six keywords were defined and a set of inclusion and exclusion criteria were established, as shown in Table 3.1.

The LWLR was divided into two Steps: 1) using the keywords in searching for work on the search engines and 2) applying the inclusion and exclusion criteria to the obtained works. In Step One, the keywords were used in a query format, which was: "Low-Code" OR "Low-Code Development" OR "Low-code Platform" OR "Low-Code Development Platform" OR "Security in Low-Code" OR "Security in Low-Code Development". Thereafter, in Step Two, the inclusion and exclusion criteria were applied to obtain a precise result that is related to the topic of this work. Table 3.2 shows the consulted search engines and repositories for this effect, together with the number of works found on each one.

Table 3.2. Search engines and works found in each one for Steps One and Two

Search Engine	Number of works found in Step 1	Number of works found in Step 2
ACM	793	323
Google Scholar	14000	8060
IEEE	510	179
Research Gate	83	59
Springer	974	77

3.2.2. Database Search

To gain a better insight into the platforms' vulnerabilities, a search was conducted on CVE Details, a database with information regarding vulnerabilities across multiple applications. For this purpose, the second path of the approach was conducted, as shown in Figure 3.1.

Therefore, firstly, a list of LCDPs was raised, based on a search done on three different sources: G2 website [**44**], Gartner website [**45**] and Magic Quadrant [**23**]. Table C.1, present in Appendix C, shows all the LCDPs found and in which source it was found. With this list, all platforms were searched on the CVE Details database, to get the vulnerabilities that each of the platforms had. Despite searching for all of the platforms on the database, not all of them were present, as shown in Table 3.3. Table 3.3 presents the platforms that were present in CVE Details and the weights that were given to each one. Based on Figure 3.2, the platforms present in the top-right quadrant were given a weight of ten, the platforms in the top-left and bottom-right quadrants were given five, and the rest were given only one value of weight. This weight system was based on Gartner's Magic Quadrant for Enterprise Low-Code Application Platforms [**24**], published in December 2022. The platforms that were not available in CVE Details were not taken into consideration since there was no information regarding their vulnerabilities. Possible causes for this might be the early age of the platform making them unknown on the market, the vulnerabilities that have not yet been reported or the absence of vulnerabilities on the platform so far.

After getting all the vulnerabilities' data, an Excel table was created on which the information was put and metrics such as raw data average and maximum score were calculated. Table C.2, present in Appendix C, presents all the information regarding the vulnerabilities available on the LCDPs presented in Table 3.3.

Table C.2 shows seven columns: the name of the platform, the CWE-ID that identifies the vulnerability, the date on which the vulnerability was last updated, the score of the vulnerability, the weight given to each platform, the raw average of all of the scores from the vulnerabilities of each platform, and the maximum score of a vulnerability for each platform. The CWE-ID is a unique code that identifies a vulnerability. The update date of the vulnerability is the date of the last update on the platform's vulnerability. The score

TABLE 3.3. Platforms available in CVE Details and corresponding weights

Low-Code Development Platform Name	**Weight**
1C	1
Agilepoint NX	1
Airtable	1
Appian	5
Blueprism RPA	1
Claris Filemaker	1
Decisions	1
HCL Domino	1
Intrexx	1
Joget Dx	1
Mendix	10
Openedge	1
Oracle Apex	5
OutSystems	10
Pega	5
Processmaker	1
Salesforce	10
ServiceNow	10
Wavemaker	1
Zoho	1

of each vulnerability is a value between 0 (zero) and 10 (ten) that relates to the ranking of the exposure, according to CWE [**46**]. This allows us to measure the importance of a vulnerability: 0 (zero) means that it is not important and 10 (ten) means that it is very important. The raw average of the scores is a simple average operation of all vulnerability scores of each platform, thus obtaining an average score for each platform. The maximum score is the highest score of all vulnerabilities present in each platform.

After collecting this data, all the information was grouped in categories, more specifically grouping all the vulnerabilities with the same ID. This was the second step of the second path of the approach, presented in Figure 3.1. During this process, a new table was created containing metrics using the information gathered regarding the LCDPs. Table 3.4 shows an example of the platforms on which the vulnerability CWE-269 ("Improper Privilege Management") occurred and Table 3.5 shows the metrics calculated for it. In Table 3.4, it can be seen that this vulnerability occurred on three different platforms, hence the number of occurrences being four. Because of this, the total weight of the vulnerability will be the sum of the platforms' weights. The total percentage divides the weight of CWE-269 by the sum of the weight of all vulnerabilities, thus obtaining 3,6%. Also, the sum score, the average score, and the maximum score are metrics pertaining to the sum of all scores of the vulnerability, the average of all scores of the vulnerability, and the maximum score of the vulnerability, respectively. Table 3.5 shows the metrics calculated for this step: total weight, total percentage weight, number of occurrences, sum

FIGURE 3.2. Graph from Magic Quadrant for Enterprise Low-Code Application Platforms 2022 [**24**], by Gartner

score, average score and maximum score. Total weight is calculated in the following way: every time a vulnerability occurs in a platform, we get the weight value of that platform and add to that vulnerability and do that to all platforms; after that, we end up with the sum of weights from all platforms on which that vulnerability occurred. Total percentage weight is the total weight of a specific vulnerability divided by the sum of all total weights of all platforms, giving us a percentage. Number of occurrences is the number of time that each vulnerability was present on the searched platforms. The sum score is, as the name implies, the sum of all the scores of that specific vulnerability in each platform. The average score is the sum score divided by the number of occurrences of each vulnerability. Maximum score is the highest score that each vulnerability has on all platforms.

Afterwards, there was a prioritization of the vulnerabilities in case of repeated results for a specific metric, presented as the third step in the second path of Figure 3.1. Because of the grouping of the vulnerabilities, there were still overlapping results that needed to be untied. To accomplish this, five industry experts were consulted in an interview format.

Table 3.4. Example for demonstration - Platforms

Low-Code Development Platform Name	CWE-ID	Score	Weight
Mendix	269	5,0	10
Mendix	269	6,5	10
Pega	269	7,5	5
Openedge	269	7,2	1

Table 3.5. Example for demonstration - CWE-269 ("Improper Privilege Management")

CWE-ID	Total Weighted	Total %	Nº Occurrences	Sum Score	Avg Score	Max Score
269	26	3,6%	4	26,2	6,55	7,5

3.2.3. Interviews

On the third path of Figure 3.1, interviews with three security experts and two pen-testers were designed. Firstly, the two pen-testers were interviewed, both with more than ten years of experience in the field. Thus, the interviews with the pen-testers' main goal was to get more information from people who had tested the LCDPs and could report directly from field data. These interviews took place between May and June 2023, were recorded with the respondents' consent, and lasted between 40 and 60 minutes. Due to confidentiality issues from the company, some specific results could not be discussed in depth. Despite this, the pen-testers provided anonymous data and true field data. The format of the interview was divided into two parts: first a questionnaire and second open questions. The questionnaire was developed to know the interviewees' experience and, based on it, obtain information regarding the topic of this work. Table 3.6 shows the questions for the interview with the pen-testers.

Afterwards, based on the gathered list of vulnerabilities from the database search and the results from the interviews with the pen-testers, three security experts from the industry were interviewed, with field experience between two and twenty years. These interviews were conducted during May 2023, were recorded with the respondents' consent, and lasted between 10 and 30 minutes. The format was an open discussion using a questionnaire based on our findings, and was divided into two parts: first a simple survey and second open questions. A set of questions was developed in order to obtain the experts' opinions on the created artifact. Table 3.7 shows the set of questions for the interview with the security experts.

Following the interviews, the information obtained was coded, as shown in step two of the third path of the approach, shown in Figure 3.1. During this step, the data was labelled according to the topic associated with it. Afterwards, this data was grouped into categories, as shown by the third step of path three of the approach, presented in Figure 3.1. This enabled the information to be organized in order to help identify common topics

Table 3.6. Questionnaire for the interviews with the pen-testers

Questionnaire
1) How many apps did you pen-test?
2) What kind of pen-test did you do?
3) What is each app used for?
4) Do the apps have internet or intranet access?
5) How many developers were involved in the development of each app?
6) Which programming language was used for developing the app?
7) Which vulnerabilities are more recurrent and/or more dangerous, based on your experience?
8) Have there been changes after the report on the pen-test?
9) Based on your experience, are vulnerabilities caused by developers or by the platform?
10) Based on your experience, are there any vulnerabilities on the platforms?
11) In your opinion, do you think that the vulnerabilities are related to the programming language used to develop the apps?
12) In your opinion, do you think that the vulnerabilities are related to the programming language that you can use on the platform?

Table 3.7. Questions for the interviews with the security experts

Questions
1) Present the findings from the database search and pen-tester interviews.
2) Ask if the experts agree with the findings.
3) Ask what the experts would change, based on their experience and knowledge.

addressed by the interviewees. Consequently, the data was prioritized, as shown in step four of the third path, as demonstrated in Figure 3.1.

3.2.4. Expert Review

On the fourth path of Figure 3.1, with all the gathered information, we asked three experts from the industry to review it to validate, approve all research and interviews done, and validate the consolidated results. Also, we appealed to the experts to help narrow down and prioritize the list in case of double results between the different paths and to get clarification on the unknown vulnerability (CWE-?). The experts helped in better prioritizing double vulnerability results, according to their experience, and that the unknown vulnerability should be considered as not yet defined. Therefore, it was possible to create this work's artifact and, with the help of the experts, improve and extend it.

CHAPTER 4

Results

This section presents all of the results acquired through the conducted research.

Throughout the research, while getting the results, there was a need to create perspectives so that the vulnerabilities found could be assigned to that perspective for a better understanding of the LCDP vulnerabilities. These were a product of the literature review conducted and of the initial discussions with the supervisors and the experts from the industry. Therefore we specify and present our results in three perspectives: *platform*, *developer*, and *plugins*. We considered the platform perspective to be related to the vulnerabilities of the environment where the application is developed or runs, thus covering the LCDP application deployment aspect. We specify the developer perspective as the problems the LCDP developer causes or introduces to the LCDP-developed application throughout the software development life-cycle. This perspective focuses on problems generated by the developer of the application and does not consider problems incurred through the usage of external components. Lastly, we defined the plugin's perspective as the problems that may occur in the developed solution due to the inclusion of third-party components, e.g., from the LCDP plugin marketplace. These perspectives are essential for understanding what are the causes of the vulnerabilities. Table 4.1 shows the mapping of information sources between the research method and LCDP vulnerability perspectives. This mapping of information will be followed during Chapter 4 and associates an activity made during the research with the perspectives.

Table 4.1. Mapping of information sources

	LWLR	CVE Details	Interview
Platform	•	•	
Developer	•		•
Plugins	•		•

Table 4.2 shows a list of all CWE-IDs present in the results of this work and their respective vulnerability name. The vulnerabilities present in this table were the CWE-IDs that were found out of the more than 1000 CWEs defined by MITRE, making these relevant for the LCDPs and for this work. Furthermore, the number of occurrences alone is not a determinant of the importance of the vulnerability, which is the reason why extra work was done with the metric calculations that will be presented afterwards. Nevertheless, Table 4.2 is already a contribution of the work, albeit not the final artifact.

In Table 4.2, we observe that the vulnerability with the highest number of appearances is CWE-79, i.e., the *cross-site scripting* vulnerability. In the second place, we found the

TABLE 4.2. List of the CWE-IDs present in this work and its number of occurrences

CWE-ID	**Name of the Vulnerability**	**Nº Ocurrences**
79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45
89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	6
352	Cross-Site Request Forgery (CSRF)	6
20	Improper Input Validation	5
269	Improper Privilege Management	4
287	Improper Authentication	4
400	Uncontrolled Resource Consumption	4
668	Exposure of Resource to Wrong Sphere	4
918	Server-Side Request Forgery (SSRF)	4
611	Improper Restriction of XML External Entity Reference	3
200	Exposure of Sensitive Information to an Unauthorized Actor	2
284	Improper Access Control	2
326	Inadequate Encryption Strength	2
425	Direct Request ('Forced Browsing')	2
434	Unrestricted Upload of File with Dangerous Type	2
502	Deserialization of Untrusted Data	2
829	Inclusion of Functionality from Untrusted Control Sphere	2
863	Incorrect Authorization	2
22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	1
74	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')	1
94	Improper Control of Generation of Code ('Code Injection')	1
120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	1
203	Observable Discrepancy	1
209	Generation of Error Message Containing Sensitive Information	1
281	Improper Preservation of Permissions	1
310	Cryptographic Issues	1
521	Weak Password Requirements	1
522	Insufficiently Protected Credentials	1
525	Use of Web Browser Cache Containing Sensitive Information	1
601	URL Redirection to Untrusted Site ('Open Redirect')	1
640	Weak Password Recovery Mechanism for Forgotten Password	1
669	Incorrect Resource Transfer Between Spheres	1
787	Out-of-bounds Write	1
1321	Improperly Controlled Modification of Object Prototype Attributes ('Prototype Pollution')	1

unknown ID, which refers to the fact that a specific CWE-ID was not yet defined for the vulnerability in question. Thirdly, we found CWE-89 and CWE-352, with six findings each. These vulnerabilities correspond to *SQL injection* and *cross-site request forgery*, respectively. Next, we found CWE-20, *improper input validation*, with 5 appearances. Following this, we found CWE-269, CWE-287, CWE-400, CWE-668, and CWE-918, all of which with 4 findings. These vulnerabilities pertain to *improper privilege management*, *improper authentication*, *uncontrolled resource consumption*, *exposure of resource to wrong sphere*, and *server-side request forgery*, respectively. Then, we have CWE-611, *improper restriction of XML external entity reference*, with 3 occurrences. Additionally, we found CWE-200 *exposure of sensitive information to an unauthorized actor*, CWE-284 *improper access control*, CWE-326 *Inadequate encryption strength*, CWE-425 *forced browsing*, CWE-434 *unrestricted upload of file with dangerous type*, CWE-502 *deserialization of untrusted data*, CWE-829 *inclusion of functionality from untrusted control sphere*, and CWE-863 *incorrect authorization* vulnerabilities, with 2 occurrences each. Lastly, we found the following vulnerabilities with only 1 finding: CWE-22 *path traversal*, CWE-74 *injection*, CWE-94 *code injection*, CWE-120 *classic buffer overflow*, CWE-203 *observable discrepancy*, CWE-209 *generation of error message containing sensitive information*, CWE-281 *improper preservation of permissions*, CWE-310 *cryptographic issues*, CWE-521 *weak password requirements*, CWE-522 *insufficiently protected credentials*, CWE-525 *use of web browser cache containing sensitive information*, CWE-601 *open redirect*, CWE-640 *weak password recovery mechanism for forgotten password*, CWE-669 *incorrect resource transfer between spheres*, CWE-787 *out-of-bounds write*and CWE-1321 *prototype pollution*.

Table 4.3 shows all the data obtained pertaining to the LCDPs vulnerabilities. From the example presented, this table presents the metrics calculated for each of the vulnerabilities present in Table 4.2.

With this information, the was created intermediate artifact, shown in Table 4.4, which presents us with the top three most common vulnerabilities for each perspective. The information for this transitional artifact comes from the interviews with the pen-testers and the database search. It is important to note that, despite the way that the perspectives are being presented, there is no hierarchy or most important perspective compared to the other. We took the decision to create an artifact of top three vulnerabilities for each perspective because it would result in a total of nine vulnerabilities from the perspectives, but additional work can be done to extend it. Furthermore, we make sure that they are not overlapping, thus making a top nine vulnerabilities in LCDPs across all perspectives.

The results show that from the platform perspective, the collected top three LCDP vulnerabilities are: T.1-1 – *cross-site scripting*, T.1-2 – *SQL injection*, T.1-3 – *cross-site request forgery*. Regarding the developer perspective, our collected top three LCDP vulnerabilities are T.2-1 – *access control*, T.2-2 – *business logic*, and T.2-3 – *administrative features (privileges)*. Regarding the plugins perspective, our collected top three LCDP vulnerabilities are T.3-1 – *custom-made plugins and interfaces*, T.3-2 – *data breaches*, and

TABLE 4.3. Metrics calculated for each vulnerability

CWE-ID	Total Weighted	Total %	Nº Occurrences	Sum Score	Avg Score	Max Score
20	18	2,50%	5	27,8	5,6	7,8
22	1	0,14%	1	5,0	5,0	5,0
74	10	1,39%	1	3,5	3,5	3,5
79	227	31,57%	45	159,3	3,5	9,0
89	19	2,64%	6	32,0	5,3	6,5
94	10	1,39%	1	6,5	6,5	6,5
120	1	0,14%	1	10,0	10,0	10,0
200	15	2,09%	2	9,0	4,5	5,0
203	10	1,39%	1	5,0	5,0	5,0
209	1	0,14%	1	5,0	5,0	5,0
269	26	3,62%	4	26,2	6,6	7,5
281	1	0,14%	1	0,0	0,0	0,0
284	11	1,53%	2	7,5	3,8	7,5
287	8	1,11%	4	21,7	5,4	7,5
310	5	0,70%	1	5,4	5,4	5,4
326	2	0,28%	2	9,3	4,7	5,0
352	27	3,76%	6	24,7	4,1	6,8
400	25	3,48%	4	19,3	4,8	5,0
425	10	1,39%	2	8,0	4,0	4,0
434	11	1,53%	2	13,9	7,0	7,5
502	6	0,83%	2	6,5	3,3	6,5
521	1	0,14%	1	0,0	0,0	0,0
522	1	0,14%	1	0,0	0,0	0,0
525	10	1,39%	1	1,9	1,9	1,9
601	1	0,14%	1	0,0	0,0	0,0
611	21	2,92%	3	16,8	5,6	7,5
640	5	0,70%	1	4,6	4,6	4,6
668	35	4,87%	4	18,4	4,6	5,5
669	1	0,14%	1	6,5	6,5	6,5
787	1	0,14%	1	10,0	10,0	10,0
829	10	1,39%	2	8,6	4,3	4,3
863	20	2,78%	2	10,8	5,4	6,8
918	31	4,31%	4	24,3	6,1	7,5
1321	5	0,70%	1	4,3	4,3	4,3
?	133	18,50%	28	144,3	5,2	10,0
TOTAL	719					

T.3-3 – *unauthorized access to systems*. We note that, in Table 4.4, for each perspective, the three found vulnerabilities are listed according to their impact, e.g. T.1-1 has a higher impact than T.1-3.

Table B.1 shows the papers that were reviewed for this work, along with the publishing year and a short summary.

TABLE 4.4. Intermediate results on top three vulnerabilities, for each perspective

Perspective	Ref.	CWE-ID	Vulnerability Description
Platform	T.1-1	*79*	Cross-Site Scripting
	T.1-2	*89*	SQL Injection
	T.1-3	*352*	Cross-Site Request Forgery
Developer	T.2-1	284	Access Control
	T.2-2	840	Business Logic
	T.2-3	250	Administrative Features (Privileges)
Plugins	T.3-1	-	Custom-made plugins and interfaces
	T.3-2	200	Data Breaches
	T.3-3	285	Unauthorized access to systems

TABLE 4.5. Examples of answers given in the pen-testers interviews

Examples of Answers (Pen-Testers)
"(...) vulnerabilities are not related to the programming language used on the app, but the programming language might compromise the platform (...)"
" (...) vulnerabilities on the platforms are found main in CVE Details (...)"
"(...) most common vulnerabilities are Access Control, Business Logic, Administrative Features, Inclusion of third-party components (...)"
"(...) from my experience, the vulnerabilities come mainly from the developer himself, not from the platform(...)"

TABLE 4.6. Examples of answers given in the security experts interviews

Examples of Answers (Security Experts)
"I agree with this table, but I would change a thing."
"(...) put Access Control and Privileges together because they overlap (...)"
"(...) add Injection from any kind since lack of sanitization of input is really common (...)"
"(...) you have to always be careful when integrating a plugin from an untrusted source (...)"
"(...) not surprised with Access Control being present (...)"

For the pen-testers interviews, Table 4.5 shows examples of answers from their interviews. For the security experts' interviews, Table 4.6 shows examples of answers given during the interviews.

All the interviews contributed to the improvement of the artifact because they provided information that could not be obtained through other means. Therefore, we updated the artifact one step further by making changes as the experts have said, resulting in a new table. Table 4.7 shows the final artifact.

TABLE 4.7. Final results on top three vulnerabilities, for each perspective

Perspective	Ref.	CWE-ID	Vulnerability Description
Platform	T.1-1	*79*	Cross-Site Scripting
	T.1-2	*89*	SQL Injection
	T.1-3	*352*	Cross-Site Request Forgery
Developer	T.2-1	284	Access Control and Administrative Features
	T.2-2	840	Business Logic
	T.2-3	74	Injections
Plugins	T.3-1	-	Custom-made plugins and interfaces
	T.3-2	200	Data Breaches
	T.3-3	285	Unauthorized access to systems

The results show that from the platform perspective, the collected top three LCDP vulnerabilities are: T.1-1 – *cross-site scripting*, T.1-2 – *SQL injection*, T.1-3 – *cross-site request forgery*. Regarding the developer perspective, our collected top three LCDP vulnerabilities are: T.2-1 – *access control and administrative features*, T.2-2 – *business logic*, and T.2-3 – *injections*. Regarding the plugins perspective, our collected top three LCDP vulnerabilities are: T.3-1 – *custom-made plugins and interfaces*, T.3-2 – *data breaches*, and T.3-3 – *unauthorized access to systems*. We note that, in Table 4.7, for each individual perspective, the three found vulnerabilities are listed according to their impact, e.g. T.1-1 has a higher impact than T.1-3.

Pertaining to Table 4.7 and Table 4.4, some clarifications need to be made. Firstly, the vulnerabilities T.2-1, T.2-2, T.2-3, T.3-2, and T.3-3 forced a backtracking search in order to get the corresponding CWE-ID. Secondly, the vulnerabilities T.1-1, T.1-2, and T.1-3 were already known, not needing to be backtracked to get the CWE-ID to identify them. Lastly, the vulnerability T.3-1 does not have any CWE-ID associated because there was not an ID that could possibly represent it the most.

After the creation of the final artifact, experts were consulted to review all the information obtained throughout the research. For this, three experts examined all the data from graphs, tables, images and the search itself, giving no objections. Thus, concluding the design of the artifact.

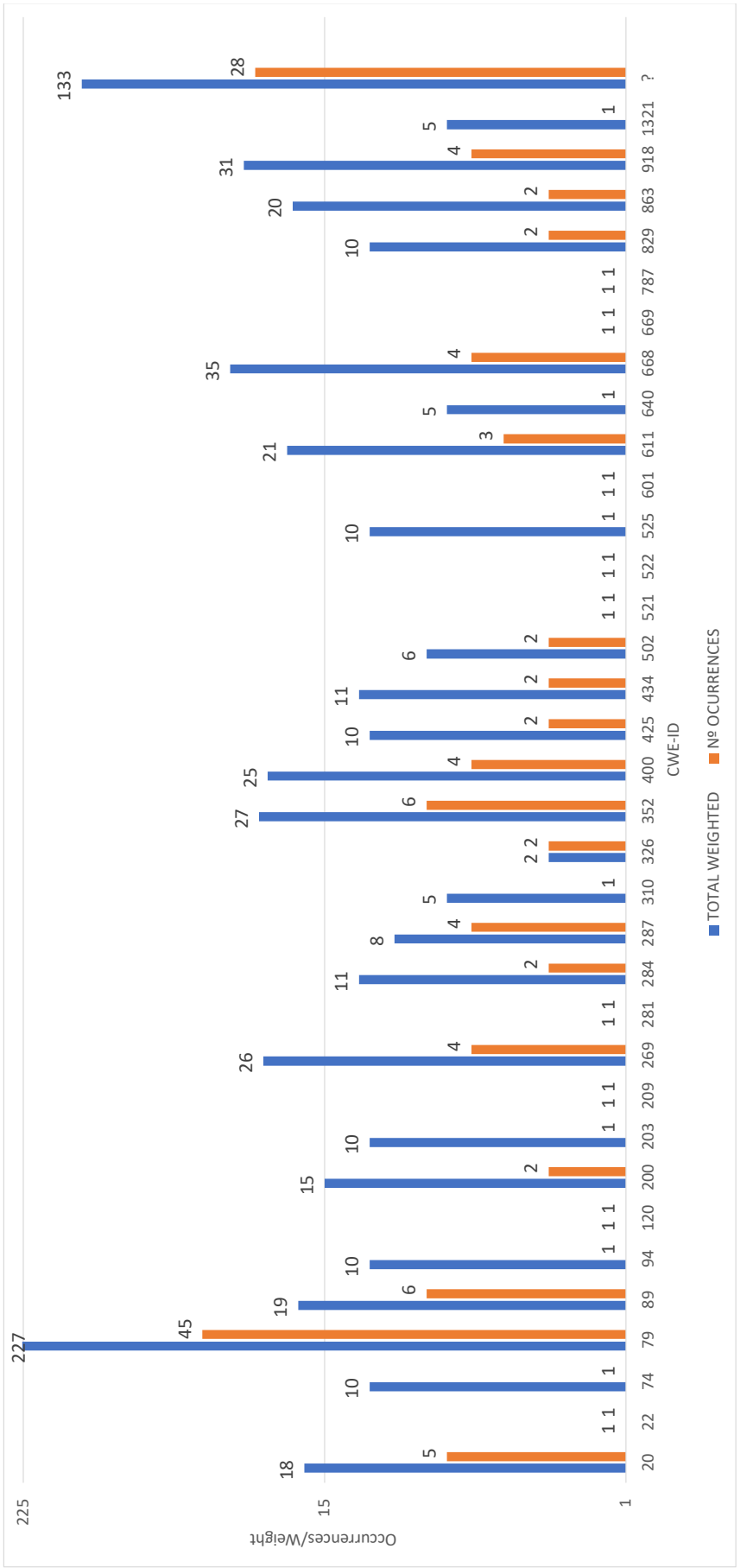

FIGURE 4.1. Comparison between Number of Occurrences and Total Weight of a vulnerability

CHAPTER 5

Discussion

The focus of this chapter is to discuss all the results obtained and presented in the previous chapter. We will address the results from the analysis done for this work.

5.1. Lightweight Literature Review

The works analyzed during the research for this work show that the scientific community covers topics related to the advantages and disadvantages of LCDPs, acceptance of LCDPs, comparison between platforms, comparison of Low-Code and code-based development, and comparison of Low-Code and model-driven approaches when developing software. This shows that this field is still young as there are recent works addressing the advantages and disadvantages of said platforms. Of course, not all platforms are equal despite all of them having the same characteristics. Because of this, the use of one platform over another has to do with the approach that the platform takes that benefits the kind of project one might develop. This does not mean that a platform is better than another, it means that each platform focuses on specific characteristics that others do not.

Also, there are few results available pertaining to the LCDPs. For example, some of the works compare market leaders but there is still limited knowledge of other platforms are less known. In the analyzed works, there were only brief mentions of the security aspect of LCDPs, which shows the immaturity of the field and a lack of research on that topic. This might have been caused by the exclusion of too many papers as some of the most recent works could have addressed the cybersecurity aspect of these platforms. But, from the analysed works, there is an absence of knowledge and approaches pertaining to the cybersecurity aspect of the platforms.

In general, the LCDPs are considered secure for software development since they automate the work for the developers. Code-based development is not 100% secure since everything is done by the developer, from the development of the software to its deployment. From the Low-Code development perspective, the developers are not always experienced in the programming field. This might also cause some security issues as they do not have the experience and knowledge to create software that complies with cybersecurity standards. Therefore, it comes to the platforms' developers to ensure that it enables a secure environment for the development of secure apps.

From this point of view, the security ends up being the responsibility of the platform's provider since they are the ones developing and finding solutions for the platforms. Despite this, the results show that there are still vulnerabilities present in the LCDPs. With this, we can conclude that platforms have problems and they need to be addressed so as not

to influence negatively the development of the software security-wise. Even though the platforms provide the possibility to add minimal lines of code, it might also bring a different class of security problems to the software.

5.2. Database Search

Concerning the research done on the CWE Details database, most of the vulnerability results were expected but some of them were a surprise.

The most recurrent and heaviest vulnerability is CWE-79, the *cross-site scripting* vulnerability. This vulnerability was one of the expected results because, as the LCDPs generally produce web apps, it only happens in a web environment. It has to do with improper sanitization of input amid the generation of the web page. There are more vulnerabilities related to lack of sanitization in the results, such as CWE-20 *improper input validation*, CWE-74 *injection*, and CWE-352 *cross-site request forgery*. These results relate to the lack of sanitization of input, leading to these vulnerabilities.

Another vulnerability that is recurrent in the results is CWE-668, the *exposure of resource to wrong sphere* vulnerability. This vulnerability was expected because the typical LCDP developer is considered to be inexperienced and not having much knowledge of programming, in other words, the typical developer is a citizen developer. This liability has to do with giving undesired actors improper access to a resource by exposing it to the incorrect control sphere. There are other vulnerabilities related to this one, such as CWE-22 *path traversal*, CWE-200 *exposure of sensitive information to an unauthorized actor*, and CWE-522 *insufficiently protected credentials*.

The vulnerabilities that were surprising were the ones related to the privilege management field, such as CWE-284 *improper access control*, CWE-269 *improper privilege management*, and CWE-287 *improper authentication*. These vulnerabilities have to do with incorrect restriction of access to a resource from an unauthorized actor. It is possible that the platform does have this vulnerability, but, generally, it is caused by citizen developers that lack cybersecurity knowledge. Hence, these liabilities were a surprising result since they generally occur in a platform.

Accordingly, follows a list of recommendations for companies and researchers for secure software development practices using LCDPs. It is vital that, before choosing the right Low-Code platform for the development of the software, a prospection on the available LCDPs be made. For this effect, the capabilities to consider on a platform are the type of apps you can build, security and quality of service, development operations (DevOps) practices, productivity and platform engineering, integration and APIs, extensibility, and reputation of the marketplace for plugins. The majority of the LCDPs have inbuilt security features including automatic testing and interoperability. Suitable LCDPs should work in conjunction with established DevOps technologies. Nonetheless, some vendors restrict options to connect with Software Development Life Cycle (SDLC) tools. Therefore, one needs to be aware of the faults and benefits to make an educated guess on which platform to use. Perhaps the goal is saving costs on software development and DevOps

is not planned at this moment; depends on the goal that one has set. The platforms should enable the integration of apps with the enterprise databases and systems in use, and simplify the use of APIs. One should also learn how to use the platforms and take advantage of their features. Most of the time, the platforms provide useful materials and features that are neglected, which through demonstrations and training programs on the platforms can help capitalize on them.

5.3. Interviews

The interviews are a fundamental part of this work. They provide a practical view of some aspects of this topic that the works and papers do not give. It is important to note that the results shown in this work were from interviews with industry experts who work for the company with whom this work was made. These interviews along with the literature review and database research complement themselves, giving both theoretical and practical information on the vulnerabilities.

From the results obtained, there is some surprising information worth addressing. Our results have shown that most of the vulnerabilities in apps developed with LCDPs are caused by the developers. As stated by the pen-testers interviewed: "Mainly, the developers are the cause of vulnerabilities on the apps." This shows that the citizen developer should be exposed to, at least, some basic security awareness workshops that are aimed at closing the gaps identified in the present work.

Another result is that some vulnerabilities pertain to third-party integrations to the project. The integration feature's goal is to facilitate development and connect with other databases and systems used by the developer. Despite this, since the majority of developers are citizen developers, liabilities like outdated libraries or custom-built interfaces are problematic. Custom-built interfaces may sometimes interfere with the platform's interface or not be able to be used in the platform itself. Outdated libraries are dangerous because very easily they can present a big threat to the app or even add unknown code that might lead to information leaks and access from outside. For example, some malware can be present in these libraries and might be fetched to a project and cause problems for the app. Hence, it is necessary to raise awareness among developers about using untrusted third-party components, which can also be done in awareness training programs.

The developed artifact in Table 4.7 gives a good guideline on which topics need to be addressed in awareness training for developers using LCDPs for software development. According to our experience, the entire workshop should be relatively short, for example lasting only a couple of hours (two or three hours). Also, according to experience in the field, these workshops could potentially contribute to significantly reducing security threats.

The conducted research and the results obtained answer the research questions defined in Chapter 1. The research questions raised for this thesis are: RQ1: What are the possible threats and vulnerabilities for Low-Code applications? RQ2: How to model risk for Low-Code software? RQ3: What are important aspects of the security life-cycle for Low-Code

applications? In order to answer RQ1, the database search done on CVE Details indicates the possible vulnerabilities for Low-Code applications are, generally, related to web apps, since they are the end product of a LCDP. Table 4.2 shows which are the vulnerabilities found during the database search process. Therefore, the vulnerabilities presented in Table 4.2 are possible threats for Low-Code applications, as of February 2023.

To respond to RQ2, we developed an approach, presented in Table 3.1, which shows the paths followed to obtain knowledge about the topic of this thesis. Each path of the approach diagram demonstrates the methodologies used to gather information. These methodologies allowed us to research different perspectives on the topic which other procedures could not, according to Table 4.1. Therefore, modelling risk for Low-Code software is possible through research and the use of methodologies that promote it.

Along with the creation and presentation of an artifact, this work also contains guidelines based on the knowledge acquired. These guidelines are recommendations for secure software development, according to the results obtained from the research and the discussion of these. Hence, this thesis also answers to RQ3.

5.4. Threats to Validity

The present work is a situated research because it was made at a specific company, which also makes the results situated. Because of this, the results and conclusions obtained might be limited and situated. Since the Design Science Research paradigm and Literature Review inspired by Kitchenham's Systematic Literature Review were used in this work, it is acceptable for industry use.

The field in which this work is positioned is still young and volatile in the industry. For this fact, the results presented in this work might change in the future, which is a problem. During the research, several paths were used to try to make the results valid, but, since this is a wicked problem, our results can change, becoming a threat to our conclusions. To counteract this, we recommend doing a periodical review of the results, as a means not to make them unusable or obsolete to the industry. Nevertheless, the results presented in this work mark the first step in understanding the vulnerabilities of LCDPs, something that has not been researched before.

For this work, the number of interviews with industry experts was small. Maybe with additional feedback, the results could have been different, which presents a threat to the results presented. However, the advantage of our results is that they are based on field data, therefore, not only theoretical but also coming from real-world use cases. Because of this, the results presented in this work can be considered relevant for other practitioners.

Although our results have some possible threats to their validity, our gained knowledge and work contribute both to the scientific body of knowledge and to the industry.

CHAPTER 6

Conclusion

Low-code development platforms constitute a new technology that is a viable alternative, especially for companies and application developers that do not have much experience in software development. Thanks to these platforms being end-user friendly, even people with little or no coding experience can develop software applications according to their ideas and requirements. Also, their ease of use through the implementation of visual features helps software development be more efficient and less time-consuming. Because of this, these platforms are gaining traction and are being considered for adoption by the industry. Our work shows that, with more convenient access to software development and the increase of citizen developers, it is necessary to raise awareness of the security aspects of these platforms. Even though the security of LCDPs is referenced in the industry and the cybersecurity experts know what the problems are, there are issues that occur due to a new development process with new challenges. With this work, we aim to understand the typical vulnerabilities present in LCDPs and in software developed with them. Our work is carried out through studies of the industry, conducting a Lightweight Literature Review on works from the industry, analysing openly known platform vulnerabilities, and interviewing six industry experts in the field. Considering that any person can develop despite their experience and knowledge of software development and programming, cybersecurity can become a problem. Our results shed light on the top three vulnerabilities of applications developed using LCDPs into three perspectives: platform, developer and plugins. We show that not only typical software development vulnerabilities can occur but also additional vulnerabilities due to the development and deployment platform itself and the inclusion of third-party plugins. This work contributes to the industry and academia, enabling the development of more secure applications, stimulating research in the field, and contributing to the cybersecurity body of knowledge. Through this work, we also provide a list of recommendations that can aid LCDP developers in improving the security of the development of their software. In future work, we intend to look at possible ways to automate the security evaluation of LCDP apps because of the dynamic nature of this field, their relevancy and their potential changes.

References

[1] OutSystems, "What Is Rapid Application Development?" (accessed in Sep. 18, 2023). [Online]. Available: https://www.outsystems.com/glossary/what-is-rapid-application-development

[2] Wikipedia, "Low-code development platform," (accessed in Sep. 18, 2023). [Online]. Available: https://en.wikipedia.org/wiki/Low-code_development_platform

[3] Kissflow, "The History of Low-Code Platforms : How Development Changed," (accessed Jul. 10, 2023). [Online]. Available: https://kissflow.com/low-code/history-of-low-code-development-platforms

[4] Forrester Research Inc., "Forrester Website," (accessed in Sep. 2, 2023). [Online]. Available: https://www.forrester.com/bold

[5] C. Richardson and J. Rymer, "New Development Platforms Emerge For Customer-Facing Applications," (accessed in Sep. 2, 2023). [Online]. Available: https://www.forrester.com/report/New-Development-Platforms-Emerge-For-CustomerFacing-Applications/RES113411

[6] Gartner Inc., "Gartner Website," (accessed in Aug. 12, 2023). [Online]. Available: https://www.gartner.com/en

[7] IBM Cloud Education, "Low-Code vs. No-Code: What's the Difference?" (accessed in Aug. 13, 2023). [Online]. Available: https://www.ibm.com/blog/low-code-vs-no-code

[8] Mendix Technology BV, "Mendix Website," (accessed in June 2, 2023). [Online]. Available: https://www.mendix.com

[9] Gartner Inc., "Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 23% in 2021," (accessed in May 6, 2023). [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2021-02-15-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-23-percent-in-2021

[10] Gartner Inc., "Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023," (accessed in May 6, 2023). [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023

[11] Gartner Inc., "Definition of Hyperautomation - Gartner Information Technology Glossary," (accessed in May 10, 2023). [Online]. Available: https://www.gartner.com/en/information-technology/glossary/hyperautomation

[12] Gartner Inc., "Becoming Composable: A Gartner Trend Insight Report," 2021, (accessed in Aug. 12, 2023). [Online]. Available: https://www.gartner.com/en/doc/becoming-composable-gartner-trend-insight-report

[13] J. den Haan, "5 Benefits of Low-Code Application Development," (accessed in Aug. 13, 2023). [Online]. Available: https://www.mendix.com/blog/benefits-low-code-development

[14] R. Roul, "32 Low-code Development Statistics to Know Before Adopting," (accessed in Sep. 21, 2023). [Online]. Available: https://www.g2.com/articles/low-code-development-statistics

[15] M. Bargury, "Major Security Breach From Business Users' Low-Code Apps Could Come in 2023, Analysts Warn," (accessed May 5, 2023). [Online]. Available: https://www.darkreading.com/edge-articles/major-security-breach-from-business-users-low-code-apps-could-come-in-2023-analysts-warn

[16] N. Liu, "Forrester: Low-Code, Citizen Development Will Lead to Major Data Breach in 2023," (accessed May 5, 2023). [Online]. Available: https://www.sdxcentral.com/articles/analysis/forrester-low-code-citizen-development-will-lead-to-major-data-breach-in-2023/2022/11/

[17] B. Kliger, "Hackers Abuse Low-Code Platforms And Turn Them Against Their Owners - Zenity | Security for Low-Code/No-Code Development," (accessed in May 12, 2023). [Online]. Available: https://www.zenity.io/hackers-abuse-low-code-platforms-and-turn-them-against-their-owners

[18] R. Ranjan, "Password Spraying Attack," (accessed Sep. 3, 2023). [Online]. Available: https://owasp.org/www-community/attacks/Password_Spraying_Attack

[19] Kaspersky Lab, "Living off the Land (LotL) attack," (accessed in May 12, 2023). [Online]. Available: https://encyclopedia.kaspersky.com/glossary/lotl-living-off-the-land

[20] M. Lourenço, T. E. Gasiba, and M. Pinto-Albuquerque, "You Are Doing it Wrong - On Vulnerabilities in Low Code Development Platforms," in *CYBER 2023, The Eighth International Conference on Cyber-Technologies and Cyber-Systems*, 2023, pp. 12–18.

[21] International Society of Automation, "ISA/IEC 62443 Series of Standards," (accessed in May 5, 2023). [Online]. Available: https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards

[22] International Organization for Standardization, "ISO/IEC 27000:2018," (accessed in May 5, 2023). [Online]. Available: https://www.iso.org/standard/73906.html

[23] Gartner Inc., "Gartner Magic Quadrant," (accessed in Aug. 12, 2023). [Online]. Available: https://www.gartner.com/en/research/methodologies/magic-quadrants-research

[24] Y. Natis, J. Hill, P. Iyengar, G. Alvarez, J. Loveland, and C. Howard, "Magic Quadrant for Enterprise Low-Code Application Platforms," 09 2021, (accessed in Sep. 29, 2023). [Online]. Available: https://www.gartner.com/en/documents/4022825

[25] OutSystems, "OutSystems Website," (accessed in June 2, 2023. [Online]. Available: https://www.outsystems.com

[26] Microsoft, "Microsoft Power Apps Website," (accessed in Sep. 1, 2023). [Online]. Available: https://powerapps.microsoft.com/en-gb

[27] ServiceNow, "ServiceNow Website," (accessed in May 10, 2023). [Online]. Available: https://www.servicenow.com

[28] Salesforce Inc., "Salesforce Website," (accessed in June 2, 2023. [Online]. Available: https://www.salesforce.com/eu/

[29] MITRE Corporation, "MITRE Website," (accessed in Aug. 12, 2023). [Online]. Available: https://www.mitre.org

[30] MITRE Corporation, "Common Vulnerabilities and Exposures (CVE) Website," (accessed in Sep. 2, 2023). [Online]. Available: https://www.cvedetails.com

[31] MITRE Corporation, "Common Weakness Enumeration (CWE) Website," (accessed in May 5, 2023). [Online]. Available: https://cwe.mitre.org/index.html

[32] OWASP Foundation, "OWASP Top Ten," (accessed in May 5, 2023). [Online]. Available: https://owasp.org/www-project-top-ten

[33] OWASP Foundation, "OWASP Website," (accessed in May 5, 2023). [Online]. Available: https://owasp.org

[34] Gartner Inc., "Definition of Citizen Developer - Gartner Information Technology Glossary," (accessed in May 6, 2023). [Online]. Available: https://www.gartner.com/en/information-technology/glossary/citizen-developer

[35] Mendix Technology BV, "Model Driven Development," (accessed in Sep. 23, 2023). [Online]. Available: https://www.mendix.com/platform/model-driven-development

[36] A. Hevner and S. Chatterjee, "Design science research in information systems," *Design research in information systems: theory and practice*, pp. 9–22, 2010.

[37] Interaction Design Foundation, "Wicked Problems," (accessed in Sep. 23, 2023). [Online]. Available: https://www.interaction-design.org/literature/topics/wicked-problems

[38] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," *Information and Software Technology*, vol. 2, 01 2007.

[39] IEEE, "IEEE Xplore Website," (accessed in May 10, 2023). [Online]. Available: https://ieeexplore.ieee.org/Xplore/home.jsp

[40] Association for Computing Machinery, "ACM Digital Library Website," (accessed in May 10, 2023). [Online]. Available: https://dl.acm.org

[41] Springer Nature, "Springer - International Publisher Website," (accessed in May 10, 2023). [Online]. Available: https://www.springer.com/gp

[42] ResearchGate GmbH, "ResearchGate Website," (accessed in May 10, 2023). [Online]. Available: https://www.researchgate.net

[43] Google, "Google Scholar Website," (accessed in May 10, 2023). [Online]. Available: https://scholar.google.com

[44] G2.com, "Top Free Low-Code Development Platforms," (accessed in May 6, 2023). [Online]. Available: https://www.g2.com/categories/low-code-development-platforms/free

[45] Gartner Inc., "Enterprise Low-Code Application Platforms Reviews and Ratings," (accessed in May 6, 2023). [Online]. Available: https://www.gartner.com/reviews/market/enterprise-low-code-application-platform

[46] Forum of Incident Response and Security Teams, "Common Vulnerability Scoring System version 3.1: Specification Document," (accessed in August 12, 2023). [Online]. Available: https://www.first.org/cvss/specification-document

[47] F. Sufi, "Algorithms in Low-Code-No-Code for Research Applications: A Practical Review," *Algorithms*, vol. 16, no. 2, 2023. [Online]. Available: https://www.mdpi.com/1999-4893/16/2/108

[48] J. Cabot and R. Clarisó, "Low Code for Smart Software Development," *IEEE Software*, vol. 40, no. 1, pp. 89–93, 2023.

[49] D. Di Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer, "Low-Code Development and Model-Driven Engineering: Two Sides of the Same Coin?" *Softw. Syst. Model.*, vol. 21, no. 2, pp. 437–446, 2022.

[50] A. Trigo, J. Varajão, and M. Almeida, "Low-Code Versus Code-Based Software Development: Which Wins the Productivity Game?" *IT Professional*, vol. 24, no. 5, pp. 61–68, 2022.

[51] A. Bucaioni, A. Cicchetti, and F. Ciccozzi, "Modelling in Low-Code Development: A Multi-Vocal Systematic Review," *Softw. Syst. Model.*, vol. 21, no. 5, pp. 1959–1981, 2022.

[52] S. Käss, S. Strahringer, and M. Westner, "Practitioners' Perceptions on the Adoption of Low Code Development Platforms," *IEEE Access*, vol. 11, pp. 29 009–29 034, 2023.

[53] J. Kirchhoff, N. Weidmann, S. Sauer, and G. Engels, "Situational Development of Low-Code Applications in Manufacturing Companies," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '22, 2022, p. 816–825. [Online]. Available: https://doi.org/10.1145/3550356.3561560

[54] D. Pinho, A. Aguiar, and V. Amaral, "What About the Usability in Low-Code Platforms? A Systematic Literature Review," *Journal of Computer Languages*, vol. 74, p. 101185, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S259011842200082X

[55] F. Khorram, J.-M. Mottu, and G. Sunyé, "Challenges & Opportunities in Low-Code Testing," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering*

Languages and Systems: Companion Proceedings, ser. MODELS '20, New York, NY, USA, 2020. [Online]. Available: https://doi.org/10.1145/3417990.3420204

[56] A. Sahay, A. Indamutsa, D. Di Ruscio, and A. Pierantonio, "Supporting the Understanding and Comparison of Low-Code Development Platforms," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2020, pp. 171–178.

APPENDIX A

Time Graph

Figure A.1 defines the activities that took place during the research for this thesis. All the activities started during my time in Munich, except the Literature Review which started in September 2022. There were no activities dependent on the other ones, giving the possibility to execute multiple tasks at the same time. The activities were approved by the supervisors and closely monitored by them.

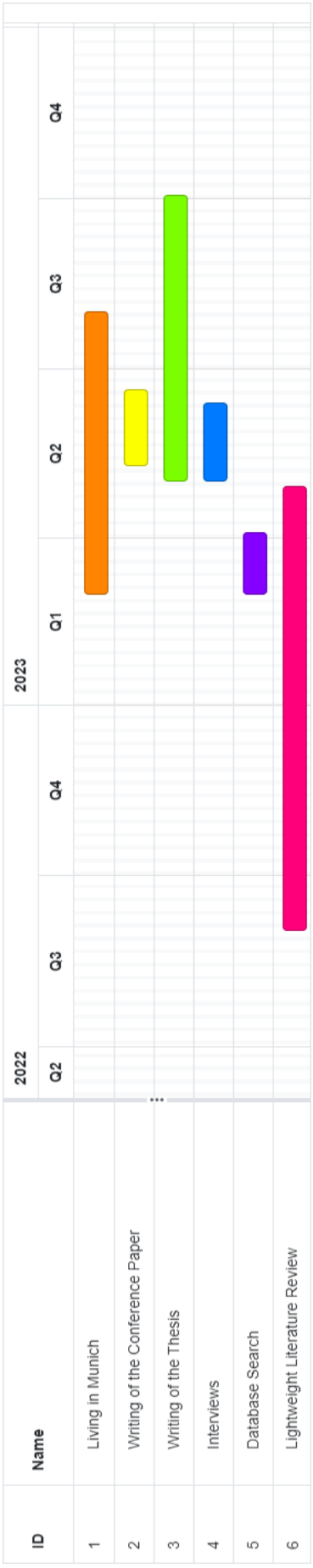

FIGURE A.1. Time Graph of all Activities

APPENDIX B

Works Analysed

Table B.1 shows a list of read works that contributed to the writing of this thesis. Each work is also followed by the date of publication of the work and a summary of the topic addressed in it.

Table B.1: List of reviewed articles from Lightweight Literature Review

Title	**Year**	**Reference**	**Short Summary**
Algorithms in Low-Code-No-Code for Research Applications: A Practical Review	2023	[**47**]	This work gives us information about the advantages and downsides of the LCDPs, supported by some examples. It also shows how to create artificial intelligence (AI) without coding, followed by an example of an algorithm that monitors cyber-attacks through a LCDP.

Continued on next page

Table B.1: List of reviewed articles from Lightweight Literature Review (Continued)

Title	Year	Reference	Short Summary
Low Code for Smart Software Development	2022	[**48**]	In this article, the authors explore the potential and challenges of low-code environments, which enable quick delivery of AI-enhanced software solutions, and provide a "wish list" for developers to consider in these tools.
Low-code development and model-driven engineering: Two sides of the same coin?	2022	[**49**]	This expert-voice paper compares low-code and model-driven approaches, identifying differences, commonalities, strengths, and weaknesses, and suggests cross-pollination directions.

Continued on next page

Table B.1: List of reviewed articles from Lightweight Literature Review (Continued)

Title	Year	Reference	Short Summary
Low-Code Versus Code-Based Software Development: Which Wins the Productivity Game?	2022	[**50**]	This article presents an experiment comparing low-code and code-based software development technologies, aiming to answer which technology enhances productivity. Results show clear productivity gains can be achieved using low-code technology in management information system development. The article reviews concepts, methodology, results, discussion, and limitations and suggests future research.

Continued on next page

Table B.1: List of reviewed articles from Lightweight Literature Review (Continued)

Title	Year	Reference	Short Summary
Modeling in low-code development: a multi-vocal systematic review	2022	[**51**]	This article presents a systematic review of low-code development, focusing on its relationship with model-driven engineering. The article, based on 58 primary studies, provides a comprehensive snapshot of low-code development during its peak of inflated expectations technology adoption phase.

Continued on next page

Table B.1: List of reviewed articles from Lightweight Literature Review (Continued)

Title	Year	Reference	Short Summary
Practitioners' Perceptions on the Adoption of Low Code Development Platforms	2022	[**52**]	In this work, a study was conducted in which 17 experts identified 12 drivers and 19 inhibitors for LCDP adoption. The consensus was that these factors are crucial, but the ranking is context-dependent. The study validates these factors, adds six new drivers and six new inhibitors to the knowledge, and analyzes their importance.

Continued on next page

Table B.1: List of reviewed articles from Lightweight Literature Review (Continued)

Title	Year	Reference	Short Summary
Situational development of low-code applications in manufacturing companies	2022	[**53**]	This paper presents an initial version of a situational software development method for manufacturing companies, enabling low-code application development. The method can be customized based on application requirements, low-code platform features, and team characteristics. Feedback from expert interviews supports the method's usefulness.

Continued on next page

Table B.1: List of reviewed articles from Lightweight Literature Review (Continued)

Title	Year	Reference	Short Summary
What about the usability in low-code platforms? A systematic literature review	2022	[**54**]	In this article, the authors performed a Systematic Literature Review procedure on the usability of LCDPs to understand the advantages and disadvantages of these platforms. Also, in their work, they point out that the drag-and-drop feature and end-user ability to develop software are among the characteristics more commonly mentioned in literature.

Continued on next page

Table B.1: List of reviewed articles from Lightweight Literature Review (Continued)

Title	Year	Reference	Short Summary
Challenges & Opportunities in Low-Code Testing	2020	[55]	This paper analyzes five commercial Low-Code Development Platforms (LCDP) testing components to present business advancements in low-code testing. It proposes a feature list for low-code testing, a baseline for comparison, and a guideline for building new ones. Challenges include the role of citizen developers, high-level automation, and cloud testing.

Continued on next page

Table B.1: List of reviewed articles from Lightweight Literature Review (Continued)

Title	Year	Reference	Short Summary
Supporting the understanding and comparison of low-code development platforms	2020	[**56**]	The authors worked on a technical review comparing eight representative LCDPs' characteristics and a short report on the experience of using each one. They conclude a set of features covering functionalities and each platform's services. This work aims to raise the understanding of how LCDPs can cover user requirements.

APPENDIX C

Platforms Found and Metrics

Table C.1 shows a list of all the platforms found in the research. It also shows which searched websites had a reference to each platform.

Table C.1: Platforms found for the database search

Low-Code Development Platform Name	**Where was it found?**
1C:Enterprise	G2, Magic Quadrant
4D	G2
8base	G2
AgilePoint NX	G2, Magic Quadrant, Gartner
Airkit	G2
Airtable	G2, Magic Quadrant, Gartner
Alibaba	Magic Quadrant
Alpha Anywhere	G2
Amoga	G2
App Builder	G2
App Sheet	Gartner
Appery.io	G2
Appian	G2, Magic Quadrant, Gartner
AppPlatform	Gartner
Astro Zero	Gartner
AuraQuantic	Gartner
ArcGIS AppStudio	G2
AutomationEdge	G2
Axpert Low Coding Platform	G2
Back4app	G2
Betty Blocks Platform	G2, Gartner
Bizagi	Gartner

Continued on next page

Table C.1: Platforms found for the database search (Continued)

Low-Code Development Platform Name	**Where was it found?**
Bryj	G2
BRYTER	Gartner
Bubble	Gartner
Build	Gartner
Caspio	G2, Gartner
Claris FileMaker	G2, Magic Quadrant
ClickPaaS Platform	Gartner
Comidor	G2
Convertigo	G2
cplace	G2
Creatio	G2, Magic Quadrant
Cyberium	G2
dbFront	G2
Decisions	G2, Magic Quadrant
DefinesysCloud	Gartner
Descope	G2
Draftbit	G2
DronaHQ	G2, Gartner
Eccentex AppBase	G2
EdgeReady Cloud	G2
Fielda	G2
Fliplet	G2
Flowable	G2
FlowWright	G2
FOEX Plugin Framework	G2
Formidable Forms	G2
Frontegg	G2
GeneXus	G2, Gartner
Graphite Studio	G2
Google App Maker	Gartner

Continued on next page

Table C.1: Platforms found for the database search (Continued)

Low-Code Development Platform Name	**Where was it found?**
HCL Domino	G2, Magic Quadrant
Huawei	Magic Quadrant
InRule	G2
Integrify	G2
Interfacing	G2
Intrexx	G2, Magic Quadrant
Jiandaoyun	Gartner
Jitterbit	G2
JobRouter	G2
Joget	G2, Magic Quadrant, Gartner
JourneyApps	G2
Kintone	Magic Quadrant, Gartner
Kissflow	G2, Gartner
Knack	G2
Kony Quantum (Formerly Kony App Platform)	G2
Linx	G2
Mendix	G2, Magic Quadrant, Gartner
metaphactory	G2
MobileFrame	Gartner
Neptune DXP	G2, Gartner
Neutrinos Platform	G2, Gartner
NewgenONE Digital Transformation Platform	Magic Quadrant, Gartner
Ninox	G2
OnBase	G2
OneBlink LcS	G2
Openedge	Magic Quadrant
Oracle Apex	Magic Quadrant, Gartner
Oreops	G2
OutSystems	G2, Magic Quadrant, Gartner
Pega Platform	G2, Magic Quadrant, Gartner

Continued on next page

Table C.1: Platforms found for the database search (Continued)

Low-Code Development Platform Name	**Where was it found?**
Pipefy	G2
PixieBrix	G2
PowerApps	Magic Quadrant, Gartner
PortalCMS	G2
PROCESIO	G2
ProcessMaker	G2, Magic Quadrant
Progress OpenEdge	G2
Progress Telerik	G2
ProntoForms	Gartner
Prophecy	G2
Qntrl	G2
Quickbase	G2, Magic Quadrant, Gartner
Quixy	G2, Gartner
RAD Platform	Gartner
Ready_	G2
Reify	G2
Retool	G2, Magic Quadrant, Gartner
Rintagi	G2
RunMyProcess	G2
Salesforce	Magic Quadrant, Gartner
Scheer PAS	G2
ServiceNow	Magic Quadrant, Gartner
Servoy	G2
Simplifier	G2
Skuid	G2
Slingr	G2
SMARTS Decision Management Platform	G2
SS&C Blue Prism Intelligent Automation Platform	G2, Magic Quadrant
Studio Creatio	G2, Gartner

Continued on next page

Table C.1: Platforms found for the database search (Continued)

Low-Code Development Platform Name	**Where was it found?**
Superblocks	G2
SYDLE ONE	G2
TeamDesk	G2
ToolJet	G2
TrackVia	G2, Gartner
Turbo Forms	G2
Twixl	G2
UI Bakery	G2
UiPath: Robotics Process Automation (RPA)	G2
Unqork	Magic Quadrant, Gartner
Veritone Automate Studio	G2
Vinyl	G2, Gartner
VisionX	G2
Visual LANSA	G2
WaveMaker	G2, Magic Quadrant
WEBCON BPS	G2, Gartner
WEM	Gartner
WeWeb	G2
Xojo	G2
Xpoda	G2
Zoho Creator	G2, Magic Quadrant, Gartner
Zvolv	G2, Gartner

Table C.2 presents all the data obtained from the search of all the platforms through CVE Details. It provides information on the LCDP name, the CWE-ID of the vulnerability, the update date of the vulnerability on CVE Details, the score given to the vulnerability, the weight given to each platform, the raw data average of the scores of all the vulnerabilities from each platform and the maximum score of a vulnerability in each platform.

Table C.2: Metrics calculated for the platforms

LCDPs	CWE-ID	Update Date	Score	Weight	Raw Data Average	Maximum Score
Mendix	284	22/02/2023	0,0	10	4,22	6,80
Mendix	74	20/07/2022	3,5	10		
Mendix	269	19/07/2022	5,0	10		
Mendix	200	12/07/2022	5,0	10		
Mendix	668	13/05/2022	4,9	10		
Mendix	668	19/04/2022	4,0	10		
Mendix	668	11/03/2022	4,0	10		
Mendix	863	12/11/2021	4,0	10		
Mendix	863	12/11/2021	6,8	10		
Mendix	525	12/11/2021	1,9	10		
Mendix	269	22/04/2021	6,5	10		
Mendix	918	11/09/2019	5,0	10		
OutSystems	79	08/09/2021	4,3	10	5,00	6,40
OutSystems	918	21/04/2021	5,0	10		
OutSystems	434	04/12/2020	6,4	10		
OutSystems	352	06/03/2020	4,3	10		
Salesforce	89	12/01/2023	0,0	10	5,36	7,50
Salesforce	611	12/08/2021	5,0	10		
Salesforce	611	01/04/2021	7,5	10		
Salesforce	918	01/04/2021	7,5	10		
Salesforce	?	01/04/2021	7,5	10		
Salesforce	400	12/06/2019	5,0	10		
Salesforce	20	31/10/2018	5,0	10		
ServiceNow	79	23/01/2023	0,0	10	2,64	6,50
ServiceNow	79	26/08/2022	0,0	10		
ServiceNow	79	26/08/2022	0,0	10		
ServiceNow	203	22/02/2022	5,0	10		
ServiceNow	79	12/05/2020	3,5	10		

Continued on next page

Table C.2: Metrics calculated for the platforms (Continued)

LCDPs	CWE-ID	Update Date	Score	Weight	Raw Data Average	Maximum Score
ServiceNow	94	05/10/2018	6,5	10		
ServiceNow	79	10/04/2018	3,5	10		
Appian	20	08/08/2017	7,8	5	7,80	7,80
Pega	502	07/11/2022	0,0	5	3,99	7,50
Pega	352	23/08/2022	0,0	5		
Pega	79	23/08/2022	0,0	5		
Pega	79	23/08/2022	0,0	5		
Pega	?	01/08/2022	0,0	5		
Pega	287	29/07/2022	7,5	5		
Pega	?	25/04/2022	4,0	5		
Pega	640	03/02/2022	4,6	5		
Pega	425	01/01/2022	4,0	5		
Pega	425	01/01/2022	4,0	5		
Pega	269	23/04/2021	7,5	5		
Pega	79	17/12/2020	4,3	5		
Pega	79	13/11/2020	4,3	5		
Pega	79	20/08/2020	3,5	5		
Pega	?	19/08/2020	7,5	5		
Pega	79	30/04/2020	6,8	5		
Pega	79	30/04/2020	6,0	5		
Pega	79	30/04/2020	6,0	5		
Pega	668	19/12/2019	5,5	5		
Pega	79	08/09/2017	4,3	5		
Pega	200	08/09/2017	4,0	5		
Oracle Apex	79	03/02/2023	4,3	5	4,81	10,00
Oracle Apex	400	08/12/2022	5,0	5		
Oracle Apex	79	08/12/2022	3,5	5		
Oracle Apex	79	08/12/2022	3,5	5		

Continued on next page

Table C.2: Metrics calculated for the platforms (Continued)

LCDPs	CWE-ID	Update Date	Score	Weight	Raw Data Average	Maximum Score
Oracle Apex	79	07/11/2022	4,3	5		
Oracle Apex	79	07/11/2022	4,3	5		
Oracle Apex	79	07/11/2022	4,3	5		
Oracle Apex	79	04/11/2022	4,3	5		
Oracle Apex	79	05/10/2022	3,5	5		
Oracle Apex	79	12/09/2022	4,3	5		
Oracle Apex	79	12/09/2022	4,3	5		
Oracle Apex	400	12/05/2022	5,0	5		
Oracle Apex	79	27/04/2022	4,3	5		
Oracle Apex	79	25/04/2022	3,5	5		
Oracle Apex	1321	06/04/2022	4,3	5		
Oracle Apex	400	28/03/2022	4,3	5		
Oracle Apex	829	01/03/2022	4,3	5		
Oracle Apex	79	28/02/2022	3,5	5		
Oracle Apex	79	28/02/2022	3,5	5		
Oracle Apex	79	02/12/2021	4,3	5		
Oracle Apex	829	01/12/2021	4,3	5		
Oracle Apex	?	21/07/2021	4,9	5		
Oracle Apex	?	23/10/2020	4,9	5		
Oracle Apex	?	23/10/2020	4,9	5		
Oracle Apex	?	23/10/2020	4,9	5		
Oracle Apex	?	22/10/2020	4,9	5		
Oracle Apex	?	22/10/2020	4,9	5		
Oracle Apex	79	21/07/2020	3,5	5		
Oracle Apex	?	21/07/2020	3,5	5		
Oracle Apex	?	21/07/2020	3,5	5		
Oracle Apex	?	21/07/2020	3,5	5		
Oracle Apex	?	20/07/2020	4,9	5		

Continued on next page

Table C.2: Metrics calculated for the platforms (Continued)

LCDPs	**CWE-ID**	**Update Date**	**Score**	**Weight**	**Raw Data Average**	**Maximum Score**
Oracle Apex	79	20/07/2020	3,5	5		
Oracle Apex	?	20/07/2020	3,5	5		
Oracle Apex	?	17/07/2020	4,9	5		
Oracle Apex	?	15/04/2020	4,9	5		
Oracle Apex	?	03/10/2019	5,8	5		
Oracle Apex	?	17/10/2018	10,0	5		
Oracle Apex	79	17/10/2018	9,0	5		
Oracle Apex	79	17/10/2018	4,3	5		
Oracle Apex	89	16/10/2018	6,0	5		
Oracle Apex	79	16/10/2018	4,3	5		
Oracle Apex	?	15/10/2018	7,5	5		
Oracle Apex	?	11/10/2018	10,0	5		
Oracle Apex	?	11/10/2018	5,5	5		
Oracle Apex	?	01/09/2017	5,8	5		
Oracle Apex	?	01/09/2017	5,0	5		
Oracle Apex	?	29/07/2017	5,5	5		
Zoho	79	29/10/2020	3,5	5	5,18	6,80
Zoho	352	28/08/2019	6,8	5		
Zoho	79	28/08/2019	4,3	5		
Zoho	352	29/07/2022	6,8	5		
Zoho	79	29/07/2022	4,3	5		
Zoho	310	03/10/2014	5,4	5		
Claris Filemaker	611	23/11/2021	4,3	1	4,50	4,60
Claris Filemaker	287	13/02/2020	4,6	1		
Claris Filemaker	287	13/02/2020	4,6	1		
Airtable	522	02/12/2022	0,0	1	0,00	0,00

Continued on next page

Table C.2: Metrics calculated for the platforms (Continued)

LCDPs	CWE-ID	Update Date	Score	Weight	Raw Data Average	Maximum Score
Blueprism RPA	669	24/08/2020	6,5	1	6,50	6,50
Processmaker	281	15/11/2022	0,0	1	5,20	6,50
Processmaker	89	07/06/2022	6,5	1		
Processmaker	89	07/06/2022	6,5	1		
Processmaker	89	19/04/2022	6,5	1		
Processmaker	502	19/04/2022	6,5	1		
Wavemaker	918	21/02/2019	6,8	1	6,80	6,80
HCL Domino	352	07/11/2022	0,0	1	3,88	10,00
HCL Domino	?	07/11/2022	0,0	1		
HCL Domino	?	20/09/2022	4,6	1		
HCL Domino	521	01/09/2022	0,0	1		
HCL Domino	601	01/09/2022	0,0	1		
HCL Domino	79	01/09/2022	0,0	1		
HCL Domino	20	02/11/2021	5,0	1		
HCL Domino	209	21/07/2021	5,0	1		
HCL Domino	287	21/07/2021	5,0	1		
HCL Domino	79	22/12/2020	4,3	1		
HCL Domino	787	16/12/2020	10,0	1		
HCL Domino	120	04/12/2020	10,0	1		
HCL Domino	20	01/12/2020	5,0	1		
HCL Domino	20	01/12/2020	5,0	1		
HCL Domino	326	10/07/2020	4,3	1		
1C	326	12/07/2022	5,0	1	5,00	5,00
Intrexx	79	19/10/2020	4,3	1	5,37	7,50
Intrexx	434	13/02/2020	7,5	1		
Intrexx	79	09/10/2018	4,3	1		

Continued on next page

Table C.2: Metrics calculated for the platforms (Continued)

LCDPs	**CWE-ID**	**Update Date**	**Score**	**Weight**	**Raw Data Average**	**Maximum Score**
Agilepoint NX	89	14/07/2022	6,5	1	6,50	6,50
Joget Dx	79	06/01/2023	0,0	1	1,17	3,50
Joget Dx	79	24/12/2022	0,0	1		
Joget Dx	79	29/03/2022	3,5	1		
Openedge	269	10/05/2022	7,2	1	6,80	7,50
Openedge	?	16/10/2018	7,5	1		
Openedge	284	22/11/2017	7,5	1		
Openedge	22	05/10/2015	5,0	1		
Decisions	352	09/06/2016	6,8	1	6,80	6,80