You Are Doing it Wrong - On Vulnerabilities in Low Code Development Platforms

Miguel Lourenço
Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR
Lisbon, Portugal
email: miguel_ponte@iscte-iul.pt

Tiago Espinha Gasiba
T CST SEL-DE
Siemens AG
Munich, Germany
email: tiago.gasiba@siemens.com

Maria Pinto-Albuquerque
Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR
Lisbon, Portugal
email: maria.albuquerque@iscte-iul.pt

***Abstract*—Low-Code Development Platforms (LCDPs) are gaining more and more traction, even in the industrial context, as a means for anyone with less coding experience to develop and deploy applications. However, little is known about the vulnerabilities resulting from this new software development model. This paper aims to understand vulnerabilities in applications developed and deployed on these platforms. We show that these vulnerabilities can be considered from three perspectives: platform, developer, and plugins. We determine the top three vulnerabilities for each perspective based on a review of the literature and expert interviews. Our results contribute to understanding LCDP applications' security and raise awareness of industry practitioners by providing typical LCDP security pitfalls.**

***Keywords–low code*; *software development*; *web applications*; *cybersecurity*; *industry*; *low code development platforms*; *vulnerabilities*.**

I. INTRODUCTION

Low-Code Development Platforms [1], a relatively new technology to develop software, trace their roots to software development tools from the 1990s and early 2000s. These platforms allow applications to be developed without writing code or only requiring small amounts of coding. The main idea is to enable application development for everyone - any user can quickly develop applications without needing to be a software developer or having too much knowledge about software development. Low-code development platforms bring several additional advantages compared to traditional software development. As such, developing software using LCDP is not only easier but can be more prevalent. In a 2021 study by Gartner [2], it was predicted that there would be an increase of 23% in the low-code development market due to the surge of remote development during the pandemic. Hyperautomation [3] was seen as one of the causes of the adoption of the low-code through 2022, which came to be true in the most recent study in 2023 by Gartner [4]. This latter study predicts that the low-code development technologies market will grow by 20% in 2023. The same study predicts a significant increase in the low-code application platforms, with an estimated growth of 25% during the year 2023, achieving almost $10 billion in revenue. Furthermore, the study predicts that the revenue will increase to $12 billion by 2024.

In addition to the lesser need to develop software, low-code development platforms can bring additional advantages. According to North [5], some advantages that key players in the market advertise include a shorter time to market, cost savings, an increase in productivity, easier maintenance, and support of digital transformation. The usage of LCDP is also impacting and gaining traction in industrial software development.

Bargury [6] and Liu [7] investigated cybersecurity incidents resulting from the usage of LCDP. Their work shows that cybersecurity incidents have steadily increased over the last few years. Security incidents can cause serious problems, from financial loss to loss of life, and are particularly important in the industrial context, especially in cases that affect critical infrastructure. Industrial cybersecurity standards, such as IEC 62.443 [8], provide several guidelines for the secure development of software and applications for the industry.

A study by the Department of Homeland Security (DHS) [9] calls attention to the fact that the root cause of more than 90% of cybersecurity incidents can be traced back to poor software quality. Developers can introduce these vulnerabilities through written code or by including external third-party components in products and services. While one of the main goals of LCDPs is to reduce the amount of software being developed, thus ideally reducing the number of security incidents, more understanding is needed to know about the security implications of developing applications using LCDP. In particular, there needs to be more understanding of the underlying vulnerabilities resulting from deploying and developing LCDP applications. This lack of knowledge is likely related to the fact that LCDP is a new technology.

In this work, we want to address these issues and increase our knowledge of LCDP vulnerabilities. Therefore, our work aims to generate an artifact - a list of relevant vulnerabilities that can affect applications developed and deployed using LCDPs. Our 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.

Our work contributes to industry and academia, enabling the development of more secure applications and stimulating research in the field. To the best of our knowledge, the present work is the first to address and understand the vulnerabilities and pitfalls of application development through low-code development platforms. Therefore, through the present work, industry practitioners can better understand the security pitfalls of developing and deploying applications using LCDP and thus actively address these pitfalls during the development of

 ISBN: 978-1-68558-113-8

the applications. Furthermore, the present work can serve as an additional motivation for academic research and contributes to the cybersecurity body of knowledge through empirical evidence.

Our work is structured in the following way. Section II briefly overviews previous work related to the present study. In Section III, we provide details on the research method, describe our approach to the problem, and also provide a description of our experiment setup. Our results are presented in Section IV and are discussed in detail in Section V. Finally, we conclude our work with Section VI, which provides a quick overview of our main results and details for further work.

II. RELATED WORK

This section characterizes the standards that influenced this work and discusses relevant blogs and articles found during the Lightweight Systematic Literature Review (LWLR) carried out during the research.

The IEC 62443 cybersecurity standard provides guidelines under which this research [8] is conducted. This standard aims to increase system security by reducing the number of system vulnerabilities. The increased security is achieved by specifying technical security requirements that industrial system elements must comply with. The IEC 62443 standard is especially relevant for industries that deliver products and services for critical infrastructures. One of the premises in the standard aims to identify and secure valuable system assets.

The MITRE Corporation, a US-based organization, maintains the Common Weakness and Enumeration (CWE) standard. While the MITRE Corporation drives this standard, the cybersecurity community influences it through open contributions. As of 2023, the CWE standard identifies over 1000 software vulnerability types. Although this standard aims to enumerate software vulnerabilities, it is general enough to be used in other areas of cybersecurity. Due to the lack of LCDP cybersecurity standards, in the present work, we use this standard to specify security vulnerabilities.

We conducted surveys and LWLR in the present work. Our survey design methodology finds its roots in work from Grooves et al. [10]. As input to the design of our survey, we conducted LWLR, a simplified version of the systematic literature review method by Kitchenham et al. [11].

In this method, we used a handful of keywords to search for literature about this topic, as follows: ”low-code”, ”low-code development”, ”low-code platform”, ”low-code development platform”, and ”security in low-code development”. With these keywords, we put them on five different databases: Google Scholar [12], IEEE Xplore [13], Springer Link [14], ACM Digital Library [15] and ResearchGate [16]. To conduct a selection of works, we defined a set of inclusion and exclusion parameters. The inclusion parameters are as follows: documents are single works (articles, papers, and book chapters), papers discuss LCDPs, and papers are available electronically in full-text form. The exclusion parameters are as follows: papers prior to 2020, papers not written in English, and studies conducted that do not cover LCDP. This process resulted in a small list of 10 articles listed in Table III in the appendix section.

However, these articles show a need for more knowledge regarding the security and vulnerabilities of LCDPs and applications developed using LCDPs. This issue is relevant since there has been an increase in cybersecurity incidents. As such, we expect this work to consolidate already-known information and introduce new knowledge.

III. METHODOLOGY

The present section describes the methodology followed in this work, which was used to create our artifact – a list of top LCDP vulnerabilities. This work separates the process into two focuses: the research method followed and the approach used based on the research method.

A. Research Method

For the present work, we took inspiration from the Design Science Research method by Peffers et al. [17], and Hevner et al. [18]. Based on the guidelines provided by Peffers et al. and our experience, we adopted four relevant guidelines for this research: 1) design as an artifact, 2) problem relevance, 3) rigor, and 4) contributions.

Regarding the first point, our designed artifact consists of a table of the top 3 vulnerabilities. Furthermore, this work’s problem is relevant for the industry since cybersecurity is essential in developing products and services. We aim to achieve rigor in our research by using diversified sources of information. In particular, we use existing information in databases and blog posts and validate our work through cybersecurity experts’ opinions and experience. Regarding the last guideline related to DSR methodology, our work aims to contribute to a better understanding of vulnerabilities in low-code development platforms. Additionally, the present work contributes to academia by deepening the existing knowledge on this subject.

According to our experience in cybersecurity in the industry, we decided to focus on three different perspectives to derive the top LCDP vulnerabilities: *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 lifecycle. This perspective focuses on problems generated by the developer of the application him or herself 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.

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.

 ISBN: 978-1-68558-113-8

B. Approach

To address our research goal and create our artifact, we collected data from different sources and validated the results through expert review. Figure 1 visually represents our approach. Data collection was carried out in three ways: 1) lightweight literature review, 2) database search, and 3) expert interviews. The last step (4) consisted of the consolidation of the results through expert review.

The lightweight literature review method which was followed is inspired by Kitchenham and Charters' Systematic Literature Review [11], but on a smaller scale. In particular, we did not conduct a review using the snowball method, and our selection process and reporting on the review are simplified. The following search engines and repositories were taken into consideration: IEEE [13], ACM [15], Springer [14], Research Gate [16], and Google Scholar [12]. We searched for papers with publishing dates between 2019 and 2023. The keywords used for the search were: "Low-Code", "Low-Code Development Platforms", "Security in Low-Code Platforms", and derived terms from these.

Regarding the database search, it was conducted on the Common Vulnerability and Exposures(CVE Details) [19] database, a free-of-use database on common vulnerabilities and exposures on most software available worldwide. To get a list of relevant LCDPs, we consulted the "Magic Quadrant for Enterprise Low-Code Application Platforms" provided by Gartner [20], along with the following additional sources: [21], [22]. With the list of LCDPs, we searched for any vulnerabilities for these platforms in CVE Details. From this, we obtained the vulnerabilities for each platform, their corresponding CWE ID [23] and a brief description of the vulnerability. It is essential to mention that, despite searching for all platforms, not all were in the database. Therefore, we consider the following low-code development platforms: Mendix [24], OutSystems [25], Salesforce [26], ServiceNow [27], Appian [28], Pega [29], Oracle Apex [30], Zoho [31], Claris Filemaker [32], Airtable [33], Blueprism [34], Processmaker [35], Wavemaker [36], HCL Domino [37], 1C [38], Intrexx [39], Agilepoint NX [40], Joget Dx [41], Openedge [42], Decisions [43], and Nintex [44]. Also, some vulnerabilities did not have a CWE ID, meaning they were either unmapped or unspecified. Thus, those vulnerabilities were not taken into consideration for the research. The list of vulnerabilities and affected platforms and their impact were collected in an Excel table. After obtaining this data, we grouped all vulnerabilities by their CWE ID and calculated the number of occurrences of each vulnerability across all frameworks. Finally, we ordered them by the number of occurrences and impact, and when two or more vulnerabilities had the same frequency value, we asked for security experts' opinions as a means for a tie-breaker.

Based on the gathered list of vulnerabilities, we designed a simple survey. The following process resulted from the design: 1) present our findings from database search and literature review, 2) ask if the experts agree with the findings, and 3) ask what the experts would change. We used the survey to interview six security experts in the industry. The industry professionals had diversified years of field experience, which ranged from beginners (two experts with less than five years of experience) to senior developers (four experts with more than twenty years of experience). These interviews were performed during May 2023, were recorded with the respondents' consent, and lasted between 30 and 60 minutes. The format was an open discussion using a questionnaire based on our findings. With all the gathered data, the information was coded and grouped into each perspective from the artifact, as shown in Table I. Following this, each perspective's vulnerabilities were prioritized to get a top 3 for the developer and plugins' perspective.

In the last step, all gathered data and information were reviewed by experts from the industry to validate and approve all the research and interviews done. To do this, we consulted three industry experts to validate all the information gathered. Also, we appealed to the experts to help narrow down and prioritize the list in case of double results. For example, when a double vulnerability result appeared, we consulted them to have a better prioritization, according to their experience.

Table I summarizes the mapping between the data input and the LCDP vulnerability perspective.

TABLE I
MAPPING OF INFORMATION

	LWLR	CVE Details	Interview
Platform		•	
Developer			•
Plugins	•		•

This table shows that the CVE details database mainly influences our understanding of platform vulnerabilities. Security expert interviews mainly influence our understanding of developer vulnerabilities. Finally, the lightweight literature review and the conducted security expert interviews mainly influence the understanding of plugin vulnerabilities.

IV. RESULTS

In this section, we present the main results from our research in the following sub-sections: mainly identified vulnerabilities from the CVE details database, results of experts' interviews, and final consolidated results in the form of an artifact containing the top LCDP vulnerabilities.

A. Platforms' Vulnerabilities

The bar chart in Figure 2 summarizes the ten most recurrent vulnerabilities identified in the analyzed platforms and the number of findings. We observe that the vulnerability which contains the highest number of recorded data is the CWE-79, i.e., the *cross-site scripting* vulnerability. In the second place, we found CWE-89 and CWE-352, with six findings each. These vulnerabilities correspond to *SQL injection* and *cross-site request forgery*. In the third place, we found CWE-20, with five findings, a vulnerability related to *improper input validation*. In the fourth place, with four findings each,

 ISBN: 978-1-68558-113-8

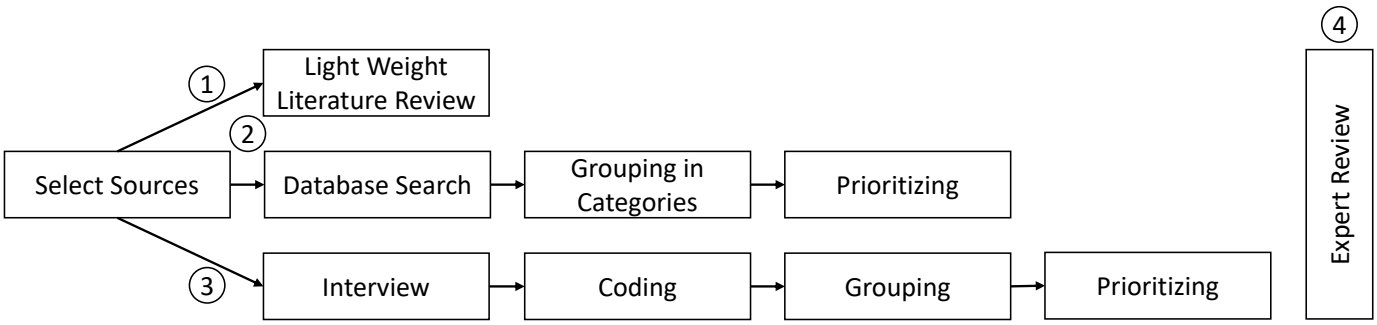

Figure 1. Process of approach for this work.

we found CWE-668, CWE-918, CWE-269, CWE-400, and CWE-287, which correspond to *resource exposure*, *server-side request forgery*, *improper privilege management*, *uncontrolled resource consumption*, and *improper authentication*, respectively. Finally, in the fifth place, we found CWE-611 with three findings corresponding to the *XML external entity* vulnerability.

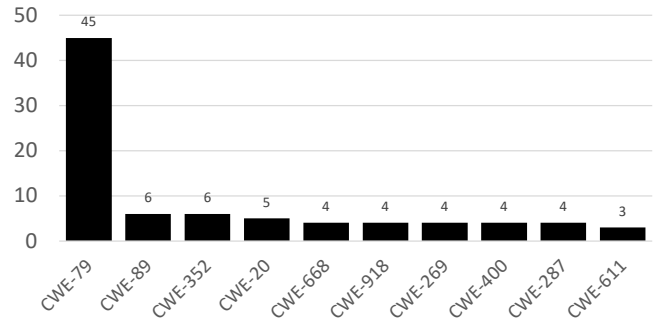

Figure 2. Identified CWE IDs together with the number of identified platform vulnerabilities from CVE details database.

B. Experts' Interview

After presenting the initial artifact, the six experts agreed with the obtained results, with few exceptions. In particular, we obtained feedback on the topic of "access control", "administrative features", and "injection vulnerabilities". The experts claimed that the two first topics could be grouped as they could be seen to overlap. Furthermore, the experts raised the point that injection vulnerabilities have not been considered, and, according to their experience, these are significant enough to belong to the list. Based on the results of the interview with the experts, not only did we validate our findings, but we could also improve and extend them.

C. Final Results

Table II shows each perspective's final results on the top vulnerabilities. The table briefly describes the vulnerability and the associated CWE ID.

Our 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 II, for each individual perspective, the three found vulnerabilities are listed according to their impact, e.g. T.1-1 has higher impact than T.1-3.

TABLE II
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	250	Injections
Plugins	T.3-1	–	Custom-made plugins and interfaces
	T.3-2	200	Data Breaches
	T.3-3	285	Unauthorized access to systems

V. DISCUSSION

This section focuses on discussing all the results obtained from the research and some possible threats to validity.

The present work considers three vulnerabilities for each perspective ordered by importance. However, we note that the order of vulnerabilities between different perspectives has yet to be considered. For example, while T.1-1 - *cross-site scripting* is the top vulnerability from the platform perspective, we do not compare its importance to T.2-1 - *access control and administrative features* of the developer perspective.

Concerning the platform's perspective, the results achieved were expected by the authors. The outcome of developing software with LCDP is a web application. As such, the results obtained according to the platforms' perspective were unsurprising, i.e., not only do they constitute typical web vulnerabilities, according to the OWASP Top 10 project [45], but they also match previous known platform incidents. Our results provide an indicator that the deployment of the LCDP platform itself should be carefully monitored, hardened, and patched. Our experience has shown that the problems present

 ISBN: 978-1-68558-113-8

in the platform perspective only occur sometimes in the applications developed within LCDP.

Regarding the developer's perspective, our results indicate that wrong configurations and implementation of business logic are the main concerns when developing LCDP applications. These results are not surprising, as the typical LCDP developer is inexperienced. However, a surprising result is the inclusion of the CWE-250 in the developer perspective. An application developed using an LCDP is typically well protected against injection attacks. However, injection problems can occur when custom components need to be developed. Thus, our results indicate that the vulnerabilities that occur using pre-defined or pre-packaged components by the LCDP vendor consist of configuration and business logic issues (T.2-1, and T.2-2). However, when custom-written components are integrated into the application, typical web development problems can occur (T.2-3).

Concerning the perspective of plugins, except for the T.3-1, the resulting findings also align with the authors' experience in the industry. The major problem we have identified is the usage of custom-made plugins and interfaces (T.3-1). This problem relates to the fact that plugins included in the LCDP application are typically custom developed, might only implement some security features, and might even lack security documentation. Therefore, custom-made plugins possess a security risk when integrated into LCDP applications without careful checking. Additionally, including externally developed plugins can lead to data leakages and data breaches, e.g., when the integrated plugin connects to an external unknown or authorized party (e.g., the plugin's vendor). This problem can lead to unauthorized access to systems (T.3-3) due to the vendor's potentially malicious usage of the components.

Finally, most analyzed platforms have a dedicated marketplace where LCDP developers can get plugins. The main idea is that the developers need not worry about security since the plugins are developed and tested by the respective vendors, and the vendors build a security stance and reputation within the marketplace. Although plugins are being vetted in the marketplace, it is still necessary to be cautious not to integrate any form of malware into the environment and projects. Acquiring third-party components through external marketplaces or specialized companies is also possible. In this case, according to our experience in the industry, it is advisable to be extra careful regarding discontinued products, obsolete versions, and malware.

A. Threats to Validity

Our lightweight literature review methodology might not have considered all existing articles on low-code-development platforms, thus potentially skewing our results. Furthermore, using the CVE Details database as the single source for the platform's vulnerabilities can bias our conclusions. Further work should therefore consider additional sources to provide a more solid validation of our results.

The present study considers a limited number of interviews with industry experts. While this small number of interviews is typical for work performed in an industrial context, this can lead to skewed and situated results. LCDPs are a new technology that, according to the author's experience, is assumed to improve and become more mature. Therefore, our results might only partially apply to current or future versions of LCDPs. As the present study is carried out in an industrial setting, it is subject to its inherent limitations in terms of the available number of experts. Nevertheless, the results obtained in the study are in agreement with the authors' experience. These results are not only corroborated by additional experts through their insightful reviews but also through practical real-world examples as obtained from feedback from the interviewed pentesters. We also note that more precise results might be obtained when the LCDP field is more mature.

Furthermore, our work summarizes vulnerabilities across different LCDPs. Due to its nature, different results might be obtained for each platform. Nevertheless, our work aims to capture an overall picture; therefore, the authors do not focus on individual platforms.

VI. CONCLUSION AND FUTURE WORK

Low-code development platforms constitute a new technology that is revolutionizing 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. 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. With this work, we study common vulnerabilities when developing and deploying applications created with LCDPs. Towards this goal, we conducted a lightweight literature review, analyzed openly known platform vulnerabilities, and interviewed six industry security experts. 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. In future work, we intend to further understand and validate our results by taking a broader approach to the topic, considering additional information from a more significant number of sources, and conducting a large-scale survey. Furthermore, the authors would like to conduct a longitudinal study approach to understand the evolution of CWE vulnerabilities in LCDPs over time. This detailed study can contribute to acknowledge on the dynamic nature of vulnerabilities, their relevancy and their potential changes.

ACKNOWLEDGEMENTS

Portuguese national funds partially finance this work through FCT–Fundação para a Ciência e Tecnologia, I.P., under the projects FCT UIDB/04466/2020 and FCT UIDP/04466/2020. Furthermore, Miguel Lourenço and Maria Pinto-Albuquerque thank the Instituto Universitário de Lisboa and ISTAR for their support.

Siemens acknowledges funding for project CONTAIN by the Federal Ministry of Education and Research under project number 13N16585.

REFERENCES

[1] M. K. Pratt, "What are Low-Code and No-Code Development Platforms?" https://www.techtarget.com/searchsoftwarequality/definition/low-code-no-code-development-platform, [Online, 2023.09.18].

[2] Gartner, "Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 23% in 2021," 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, [Online, 2023.05.10].

[3] Gartner, "Definition of Hyperautomation - Gartner Information Technology Glossary," https://www.gartner.com/en/information-technology/glossary/hyperautomation, [Online, 2023.05.10].

[4] Gartner, "Gartner Forecasts Worldwide Low-Code Development Technologies Market to Grow 20% in 2023," 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, [Online, 2023.05.10].

[5] J. North, "Will Low-Code Replace Developers?" https://www.101ways.com/is-low-code-development-a-threat-to-software-engineering/, [Online, 2023.05.10].

[6] M. Bargury, "Major Security Breach From Business Users' Low-Code Apps Could Come in 2023, Analysts Warn," https://www.darkreading.com/edge-articles/major-security-breach-from-business-users-low-code-apps-could-come-in-2023-analysts-warn, [Online, 2023.05.10].

[7] N. Liu, "Forrester: Low-Code, Citizen Development Will Lead to Major Data Breach in 2023," https://www.sdxcentral.com/articles/analysis/forrester-low-code-citizen-development-will-lead-to-major-data-breach-in-2023/2022/11/, [Online, 2023.05.10].

[8] International Electrotechnical Commission, "Understanding IEC 62443," https://www.iec.ch/blog/understanding-iec-62443, [Online, 2023.05.10].

[9] Department of Homeland Security, US-CERT, "Software Assurance," https://tinyurl.com/y6pr9v42, [Online, 2020.09.07].

[10] R. M. Groves, F. J. Fowler Jr, M. P. Couper, J. M. Lepkowski, E. Singer, and R. Tourangeau, *Survey methodology*. John Wiley & Sons, 06 2009.

[11] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," *Information and Software Technology*, vol. 2, pp. 1–65, 01 2007.

[12] Google, "Google Scholar Website," https://scholar.google.com, [Online, 2023.05.10].

[13] IEEE, "IEEE Xplore," https://ieeexplore.ieee.org/Xplore/home.jsp, [Online, 2023.05.10].

[14] Springer Nature, "Springer - International Publisher," https://www.springer.com/gp, [Online, 2023.05.10].

[15] Association for Computing Machinery, "ACM Digital Library," https://dl.acm.org, [Online, 2023.05.10].

[16] ResearchGate GmbH, "ResearchGate Website," https://www.researchgate.net, [Online, 2023.05.10].

[17] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems*, vol. 24, pp. 45–77, 01 2007.

[18] A. Hevner, S. Chatterjee, A. Hevner, and S. Chatterjee, "Design Science Research in Information Systems," *Design research in information systems: theory and practice*, pp. 9–22, 2010.

[19] MITRE Corporation, "CVE Details Website," https://www.cvedetails.com/, [Online, 2023.05.10].

[20] Gartner, "Magic Quadrant for Enterprise Low-Code Application Platforms," https://www.gartner.com/en/documents/4022825, 12 2022, [Online, 2023.05.10].

[21] Gartner, "Enterprise Low-Code Application Platforms Reviews and Ratings," https://www.gartner.com/reviews/market/enterprise-low-code-application-platform, [Online, 2023.05.10].

[22] G2.com, "Top Free Low-Code Development Platforms," https://www.g2.com/categories/low-code-development-platforms/free, [Online, 2023.05.10].

[23] MITRE, "CWE List Version 4.11," https://cwe.mitre.org/data/, [Online, 2023.05.10].

[24] Mendix, "Mendix Website," https://www.mendix.com, [Online, 2023.05.10].

[25] OutSystems, "OutSystems Website," https://www.outsystems.com, [Online, 2023.05.10].

[26] Salesforce, "Salesforce Website," https://www.salesforce.com/eu/, [Online, 2023.05.10].

[27] ServiceNow, "ServiceNow Website," https://www.servicenow.com, [Online, 2023.05.10].

[28] Appian, "Appian Website," https://appian.com, [Online, 2023.05.10].

[29] Pega, "Pega Website," https://www.pega.com/, [Online, 2023.05.10].

[30] Oracle, "Oracle APEX Website," https://apex.oracle.com/en/, [Online, 2023.05.10].

[31] Zoho, "Zoho Website," https://www.zoho.com, [Online, 2023.05.10].

[32] Claris Filemaker, "Claris Filemaker Website," https://www.claris.com/filemaker/, [Online, 2023.05.10].

[33] Airtable, "Airtable Website," https://www.airtable.com, [Online, 2023.05.10].

[34] Blueprism, "Blueprism Website," https://www.blueprism.com, [Online, 2023.05.10].

[35] Processmaker, "Processmaker Website," https://www.processmaker.com, [Online, 2023.05.10].

[36] Wavemaker, "Wavemaker Website," https://www.wavemaker.com, [Online, 2023.05.10].

[37] HCLDomino, "HCLDomino Website," https://www.hcltechsw.com/domino, [Online, 2023.05.10].

[38] 1C, "1C Website," https://1c-dn.com, [Online, 2023.06.18].

[39] Intrexx, "Intrexx Website," https://www.intrexx.com, [Online, 2023.06.18].

[40] AgilepointNX, "AgilepointNX Website," https://www.agilepoint.com, [Online, 2023.06.18].

[41] JogetDx, "JogetDx Website," https://www.joget.org/product/joget-dx/, [Online, 2023.06.18].

[42] Openedge, "Openedge Website," https://www.progress.com/openedge, [Online, 2023.06.18].

[43] Decisions, "Decisions Website," https://decisions.com, [Online, 2023.06.18].

[44] Nintex, "Nintex Website," https://www.nintex.com, [Online, 2023.06.18].

[45] Open Web Application Security Project, "OWASP Top Ten," https://owasp.org/www-project-top-ten, [Online, 2023.06.18].

[46] F. Sufi, "Algorithms in Low-Code-No-Code for Research Applications: A Practical Review," *Algorithms*, vol. 16, no. 2, 2023.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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

[53] 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

[54] 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

[55] 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.

 ISBN: 978-1-68558-113-8

APPENDIX

TABLE III
FINAL 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	[46]	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.
Low Code for Smart Software Development	2022	[47]	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	[48]	This expert-voice paper compares low-code and model-driven approaches, identifying differences, commonalities, strengths, and weaknesses, and suggests cross-pollination directions.
Low-Code Versus Code-Based Software Development: Which Wins the Productivity Game?	2022	[49]	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.
Modeling in low-code development: a multi-vocal systematic review	2022	[50]	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.
Practitioners’ Perceptions on the Adoption of Low Code Development Platforms	2022	[51]	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.
Situational development of low-code applications in manufacturing companies	2022	[52]	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.
What about the usability in low-code platforms? A systematic literature review	2022	[53]	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.
Challenges & Opportunities in Low-Code Testing	2020	[54]	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.
Supporting the understanding and comparison of low-code development platforms	2020	[55]	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.

 ISBN: 978-1-68558-113-8