Cognitive Barriers in Software Security: A Study of Novices' Interpretation of SAST Reports

Rubens Abraão da Silva Sousa
State University of Ceará
Fortaleza, Ceará, Brazil
abraao.sousa@aluno.uece.br

Gabriel Pinheiro Rezende de Lima
State University of Ceará
Fortaleza, Ceará, Brazil
gabriel.rezende@aluno.uece.br

Ismayle de Sousa Santos
State University of Ceará
Fortaleza, Ceará, Brazil
ismayle.santos@uece.br

Sávio Freire
Federal Institute of Ceará
Morada Nova, Ceará, Brazil
State University of Ceará
Fortaleza, Ceará, Brazil
savio.freire@uece.br

Abstract

Context. Static Application Security Testing (SAST) tools are widely used to identify vulnerabilities, but their effectiveness depends on users' ability to correctly interpret their reports. **Goal**. This study investigates the cognitive and technical barriers faced by novices when analyzing SQL injection and cross-site scripting (XSS) vulnerabilities. **Method**. We conducted a within-subject survey with 30 novice participants, evaluating real-world scenarios derived from the OWASP Juice Shop. We considered conceptual, contextual, terminological, and operational dimensions, as well as measures of perceived cognition. The data were quantitatively analyzed. **Results**. The results indicate that, although participants performed well on operational and technical interpretation tasks, there were significant limitations in conceptual understanding, especially in more complex scenarios. Furthermore, a misalignment between objective performance and perceived understanding was identified, highlighting metacognitive difficulties in self-assessment. **Conclusion**. The study provides evidence that barriers to the use of SAST by novices are not only technical but also cognitive, pointing to the need for improved support for understanding and explainability in security tools.

Keywords

Static Application Security Testing (SAST), Software Security, Vulnerability Analysis, Novices, Cognitive Barriers

1 Introduction

One of the major challenges in application development today is software security. The exponential growth of cataloged vulnerabilities evidences the continuous escalation of risks to critical systems, and from this scenario emerges DevSecOps, which prioritizes the integration of security protocols across all phases of the software development life cycle [9]. According to the database of the National Institute of Standards and Technology (NIST[1]), 21,452 new vulnerabilities were registered in the period from January 1, 2026, to April 27, 2026 [20], which reinforces the need for and the importance of DevSecOps.

In this context, Static Application Security Testing (SAST) play a strategic role in DevSecOps pipelines due to their ability to identify vulnerabilities, such as Cross-Site Scripting (XSS) and SQL injection, directly in source code [29].

However, the incorporation of SAST tools into continuous integration and delivery (CI/CD) cycles does not, by itself, ensure the effective resolution of detected vulnerabilities. The reports generated by these tools within the DevSecOps workflow are frequently characterized by high false-positive rates, generic alert messages, and interfaces that hinder the prioritization of genuine flaws [19, 29, 32]. These limitations produce a massive volume of low-granularity information, forcing professionals to dedicate considerable effort to the manual triage of alerts. Such a disconnection between the generated reports and the development environment interrupts the workflow, requiring practitioners to incessantly switch between different contexts to interpret, map, and remediate the reported vulnerabilities [32].

Recent studies [4, 22] have investigated cognitive load as a central factor in explaining the difficulties faced by professionals during software testing practices. Barroso et al. [4] conducted a case study with developers and identified a set of cognitive load drivers. Based on this, it is evident that the lack of adequate cognitive support compromises not only technical execution but also the contextual understanding of the artifacts involved in the testing process, suggesting the presence of conceptual barriers. In addition, Johnson et al. [12] sought to understand the challenges associated with the use of SAST tools. Through interviews with developers, Johnson et al. [12] found that the high rate of false positives is one of the primary barriers hindering the wider adoption of SAST tools within the development life cycle, being among the pioneers to address this issue.

However, the persistence of the barriers identified by Johnson et al. [12] have been noted in recent studies [5, 7, 29], particularly false positives, unclear alerts, and the effort required to decide whether a warning is actionable. Do et al. [7] focused on understanding the technical issues of SAST tools, reinforcing the negative impact of false positives generated by these tools, as well as the effort required to decide whether an alert is relevant or not. Based on these difficulties, users encounter terminological barriers due to the difficulty of understanding technical terms present in the

[1] https://www.nist.gov

alerts, and consequently, they also face contextual barriers. Wadhams et al. [29] identified, through a systematic literature review, that the barriers to the efficient use of these tools involve both technical and human obstacles. Lastly, Bennett et al. [5] surveyed 1,263 developers, finding that most respondents use SAST tools without additional configurations due to the complexity of tool setup, which suggests the presence of operational barriers.

Despite the reported advances, the literature still lacks investigations that measure the contextual, conceptual, terminological, and operational barriers specifically faced by novices when interpreting SAST reports in real-world vulnerability scenarios. This gap is concerning because novice users frequently encounter static analysis alerts without the necessary context to understand and operationalize them into effective test cases, risk analyses, mitigation plans, etc. Existing studies [4, 5, 7, 22, 29] have focused primarily on the perspective of developers without considering the level of experience in handling SAST tools, thereby neglecting the cognitive experience of novices in reading and interpreting reports.

Consequently, this gap entails relevant practical and scientific implications. Without the definition and characterization of the barriers faced by novices, it becomes unfeasible to develop and evaluate pedagogical interventions tailored to these professionals. Furthermore, it precludes the design of frameworks and cognitive support tools that assist in contextualizing reports, as well as the adequate evaluation of emerging approaches, such as the use of Large Language Models (LLMs) to aid in report interpretation.

Research in software development demonstrates that self-efficacy is a relevant predictor of performance in secure development tasks, directly influencing a professional's readiness to act when faced with complex technical artifacts [4, 22]. However, none of these dimensions have been systematically investigated in the specific context of SAST report interpretation by novices.

To address this gap, this work investigates the cognitive and technical barriers faced by novices when analyzing SQL injection and cross-site scripting (XSS) vulnerabilities. We surveyed 30 novice software practitioners (i.e., with up to three years of professional experience and who reported either no prior use or only sporadic use of SAST tools). They analyzed two vulnerability scenarios SQL Injection and XSS derived from OWASP Juice Shop and related to the OWASP Top 10[2] Injection risk category. We assessed performance across four cognitive dimensions (conceptual, contextual, terminological, and operational), alongside measures of perceived cognition and self-assessment, enabling an integrated analysis of objective performance and participant perception.

The results indicate that although participants demonstrate high performance in localization, interpretation, and execution tasks, there is a significant decline in the conceptual dimension within more complex scenarios. Furthermore, a misalignment between objective performance and perceived cognition was identified, highlighting that participants did not accurately assess their own understanding. These findings suggest metacognitive limitations and reinforces the role of conceptual complexity as a critical factor in the interpretation of SAST reports.

This study contributes to the literature by empirically highlighting the role of cognitive factors in the interpretation of security vulnerabilities, expanding the understanding of the barriers faced by novice users. From a practical standpoint, the results indicate the need for improvements in SAST tools, focusing on cognitive support and explainability.

This paper is organized as follows: Section 2 presents the concepts used in this research. Section 3 discusses the related work. Section 4 describes the research method, and Section 5 presents the results. Section 6 discusses the findings, while Section 7 addresses threats to validity. Finally, Section 8 concludes the paper and outlines future directions.

2 Background

Cognitive understanding, a concept from experimental psychology, refers to the mental processes involved in acquiring, organizing, and using knowledge and underpins software engineering activities [2]. It involves building mental representations, analogous to the transition from understanding a problem to developing a software solution, and spans activities from requirements elicitation and architectural design to debugging and evolutionary maintenance [8].

In the software development scenario, cognition manifests in the mental processes themselves, such as information encoding, storage, and retrieval. In parallel, metacognition refers to the level of monitoring, regulation, and evaluation of these cognitive processes [17]. The distinction between cognition and metacognition is operationalized in development practice: the former acts as the engine for processing technical information, while the latter functions as a means of monitoring and regulating intellectual performance, configuring a complementary action for executing complex tasks.

Metacognition, understood as the ability to think about one's own thinking, plays a central role in explaining human behavior in software engineering activities that demand effort, high performance, and the handling of complexity. Fritz et al. [10] explained the possibility of measuring mental effort and task difficulty in development through psychophysiological measures, establishing a bridge between the developer's subjective experience and the objective cognitive demand of the activity.

Regarding software development, Schoonewille et al. [24] argued that the effort perceived by developers is modulated by intrinsic, extrinsic, and germane factors, which determine the effective complexity of programming tasks. Within this same scenario, the self-perception of competence emerges as a fundamental metacognitive construct. Originating from Bandura's [3] Social Cognitive Theory, the concept of self-efficacy refers to an individual's beliefs about their capacity to organize and execute actions necessary to produce specific results [28]. Lastly, Toma and Vahrenhold [28] showed that self-efficacy directly interacts with cognitive load and emotional responses in collaborative algorithm learning, indicating that perceived competence shapes effort and persistence when facing complexity.

Cognitive Load Theory [25] distinguishes intrinsic load, arising from element interactivity and task complexity extraneous load, arising from how information is presented and germane resources devoted to constructing and refining mental schemas. In this study, Cognitive Load Theory is used as an interpretive lens rather than as a directly measured latent construct. Building on this intersection,

[2]https://owasp.org/Top10/2025/

the present study examines cognitive barriers in interpreting SAST reports, grounded in the studies discussed in this section.

3 Related Work

This section presents approaches focused on static code analysis tools, as well as studies regarding cognitive analysis within the technological context. It also highlights existing limitations in the technical literature.

Although SAST tools have evolved over the years, the effectiveness of their usage is still limited by human risk perception and efficient handling. Ami et al. [1] highlighted that one of the industry's greatest challenges is not merely the detection of vulnerabilities, but the difficulty professionals face in interpreting SAST tool reports. The study concluded that SAST tools lack better explanation mechanisms, exposing the need to overcome contextual and terminological barriers from an industry perspective.

Aligned with the perspective of Ami et al. [1], Do et al. [7] analyzed user behavior to understand the technical issues of SAST tools. As a result, they observed that software developers made decisions based on perceived importance and contextual relevance. Consequently, the cognitive effort generated by false positives proved to be a significant factor in the abandonment of tool usage, alongside the difficulty of understanding the context and technical terms of a vulnerability. However, Do et al. [7] did not perform a selection considering the proficiency and experience of the participants, leaving a gap regarding the performance of novice users when using SAST tools. Furthermore, there is a limitation concerning the impact of cognitive load arising from reading these tools' reports and the theoretical classification of the barriers encountered.

In contrast, Özkan et al. [33] showed how the introduction of Domain-Driven Design (DDD) principles impacts a real-world system. As a final result, the application of DDD showed a steep learning curve. One of the reasons for this was the terminological complexity faced by experienced professionals, evidencing that a professional's decision to act on a static analysis alert depends on the clarity of the message and the perception of difficulty. Thus, the lack of clarity in technical reports, such as those produced by SAST tools, presents itself as a relevant terminological and contextual barrier for developers.

Regarding SAST tools, Wadhams et al. [29] identified the primary barriers hindering the use these tools through a systematic literature review. The authors reported that these tools require intensive configuration that can be time-consuming and, in most cases, does not appear to be advantageous. This fact manifests as an operational barrier due to the difficulty of the initial setup necessary for the correct and efficient use of the tools. Coupled with this, they showed that the high rate of false positives is the main reason for the loss of trust in the tool, a point also emphasized by Do et al. [7]. Wadhams et al. [29] organized the difficulties in an empirical manner, yet it lacks a technical taxonomy for the efficient grouping of these barriers. However, the study did not investigate the impacts of cognitive loads that might affect the performance of novice users of these tools.

Furthermore, Barroso et al. [4] identified that cognitive load in unit testing activities is driven by communication failures and a lack of tool support, confirming the necessity of investigating the terminological and operational barriers that hinder the work of novices when dealing with SAST tools. This study used scales, validated by the authors, to measure the specific effort of professionals while performing unit tests. However, this work is limited to unit testing, lacking a more comprehensive analysis.

Lastly, Quintero-Manes and Vieira [22] validated a specific scale that seeks to differentiate cognitive load into intrinsic, extrinsic, and germane, based on Cognitive Load Theory. Through this study, the authors identified that factors, such as the quality of information received through SAST tool reports, were primarily responsible for the increase in intrinsic and extrinsic cognitive load, affecting both the professional's self-efficacy and interest. Consequently, current SAST reports impose an unnecessary extrinsic load on novice users, creating operational barriers that manifest as conceptual obstacles.

Although the technical literature [1, 7, 29, 33] has addressed the difficulties generated by the complexity of technical terms as well as operational and contextual failures, there remains a gap regarding how these barriers specifically affect novice users of SAST tools. Furthermore, the work of Quintero-Manes and Vieira [22] serves as a foundation for evaluating cognitive load, in conjunction with the characterization of technical and non-technical cognitive load drivers established by Barroso et al. [4]. The present work seeks to fill the gaps concerning cognitive barriers during the interpretation of vulnerabilities by novices through SAST reports, in addition to identifying conceptual, contextual, terminological, and operational barriers, to understand the difficulties these professionals experience when using the tool.

4 Research Method

This study aims to investigate the cognitive and technical barriers faced by novices when analyzing SQL injection and cross-site scripting (XSS) vulnerabilities. Based in this goal, we defined the following research question (RQ): *What are the primary cognitive and technical barriers that novices face when interpreting SAST tool reports (e.g., SonarQube)?* By this question, we intent to identify and measure the contextual, cognitive, terminological, and operational barriers encountered by novices while interpreting and operationalizing static analysis (SAST) reports within real-world vulnerability scenarios involving SQL Injection and Cross-Site Scripting (XSS).

These vulnerabilities are presented as among the most frequent according to the OWASP Top 10 [21] report within the Injection category. They were taken into consideration given their significance in the field and the frequency of recorded incidents. Considering their prevalence and criticality, it is observed that such vulnerabilities possess a high potential for impact, as they can compromise the confidentiality, integrity, and availability of systems; SQL Injection, for instance, allows unauthorized access to databases, while XSS enables the execution of malicious scripts within the user's browser. Furthermore, both largely stem from failures in input validation and sanitization, highlighting structural issues in software development and reinforcing their importance for academic research. Thus, the choice of these vulnerabilities is grounded not only in their incidence but also in the possibility of providing a consistent, practical analysis aligned with the current demands of the information security field.

4.1 Survey Design

To understand barriers, this study employed a survey, enabling a descriptive and cross-sectional approach [13, 18], thereby capturing a current snapshot of the novice population in the study and understanding their profile through the questions presented in Table 1. The survey was structured to include two analysis scenarios for the population. These scenarios are classified as Injection vulnerabilities, according to the OWASP Top 10 [21], specifically utilizing SQL Injection and XSS types. Vulnerabilities of this nature are exploited to inject code through the application with the aim of obstructing or exfiltrating data.

The initial version of the survey was reviewed by an experienced researcher in software testing. Based on the feedback, revisions were made to improve clarity and completeness, resulting in an updated version of the instrument. This revised version was then piloted with two participants matching the target audience (novices): one tester/QA professional and one student/intern, both with less than one year of experience and infrequent exposure to SAST reports. The pilot study aimed to evaluate the clarity of the questions, the time taken by participants, and the feedback regarding the question format. Participants identified grammatical and placeholder errors, which were subsequently corrected. Completion times were 10 minutes and 33 seconds and 9 minutes and 2 seconds, respectively; for reporting purposes, we rounded the duration to 10 minutes.

Table 1: Participant profile and characterization questions

Question	Response Options
Informed Consent Form (ICF): "I have read and agree to the terms above. I understand that my responses will be used anonymously for academic research purposes."	□ I agree
What is your primary area of expertise?	Developer Tester / QA DevOps / SRE Security Engineer / AppSec Software Architect Student / Intern
How long have you been working in software development and/or testing?	Less than 1 year 1 to 3 years 3 to 5 years 5 to 10 years More than 10 years
How often do you analyze vulnerability reports (SAST) in your routine?	Daily Weekly Monthly Rarely Never

The vulnerability scenarios are derived from a SonarQube[3] static analysis of the OWASP Juice Shop platform[4]. The choice of this platform considers its status as a global benchmark for web application security training, maintained by the organizers of the OWASP Top 10. This ensures the reproducibility of the scenarios within the survey.

With the scenarios established for replication, specific questions were developed for each, aiming to identify the study's core barriers: (i) Conceptual, the user's ability to understand the technical meaning and implications of the results provided by the security tool [27]; (ii) Operational, when the user feels unable to, or fails to find means of, transforming the result into a verifiable artifact [31]; (iii) Terminological, where the user stalls, becomes confused, or does not recognize the presented terminology [32]; and (iv) Contextual, where the user is tasked with analyzing a code context or vulnerability with which they are unfamiliar[29].

The questions were derived from the SonarQube static analysis based on the OWASP Juice Shop application code; the environment used is available at the GitHub link[5]. The criteria for considering an item as correct were established based on the vulnerabilities and guidelines of OWASP Top 10 [21] and their respective correct alternatives. The validity of the instrument's content was established through expert review. Two researchers with expertise in software security independently evaluated whether each question from Table2 aligned with its assigned dimension (conceptual, contextual, terminological, operational). Divergences were resolved by these researchers in a consensus meeting.

The survey, delivered via Google Forms, is divided into three parts: i) Demographic profile, which aimed to understand seniority, area of expertise, and experience with SAST tools in daily routines; ii) Scenarios provided in legible images, where each scenario includes four questions in the following order: 1st Conceptual, 2nd Contextual, 3rd Terminological, and 4th Operational, as shown in Table 2; iii) Likert scale, to measure the perceived Self-Declared Conceptual Domain [22]. At the conclusion of the survey, two self-assessment and confidence questions regarding the scenario responses were included, as presented in Table 3.

4.2 Data Collection

The survey was disseminated through email lists and instant messaging groups accessible to the researchers, seeking to reach participants with a profile compatible with the study's target audience. The data collection period ran from April 7 to April 25, 2026. The structure followed the questions presented in Tables 1 and 2.

Initially, the survey received 51 responses. To ensure alignment with the target profile of novice participants, inclusion criteria were applied based on professional experience (up to 3 years) and frequency of SAST tool usage (never or sporadically). After applying these criteria, the **final sample consisted of 30 participants**.

To participate in the survey, the population agreed, via the form, to sign the Informed Consent Form (ICF), acknowledging that the research data would be used solely for academic purposes.

4.3 Data Analysis

We conducted the data analysis in JASP software[6], processing the data in three formats: i) Descriptive, utilizing frequencies and measures of central tendency to analyze the collected data; ii) McNemar's Test for paired comparison between scenarios, appropriate for the binary nature of the performance data (0 = error; 1 = correct); iii) Correlation, seeking to understand the relationships between

[3]https://www.sonarsource.com/products/sonarqube/
[4]https://github.com/juice-shop/juice-shop
[5]https://github.com/ASTRID-RESEARCH/2026-SBES-Package-Replication.git
[6]https://jasp-stats.org/

Table 2: Mapping of Scenarios, Cognitive Categories, and Questions

Scenario	Category	Question	Alternatives	Correct
SC1 - SQL Injection	Conceptual	By analyzing the SAST report and the code snippet shown in this SQL injection scenario, which of the options below best describes the root cause of the vulnerability?	a) The problem is solely the lack of error handling (try/catch) b) User input data is directly concatenated into an expression/code to be executed c) The vulnerability occurs because the database query is slow d) The risk lies only in the log message	B
	Contextual	Considering the SAST report and the presented code snippet, at which point in the code do you understand that the SQL injection vulnerability effectively manifests?	a) At the moment the value is read from the form b) On the line where the input value is directly incorporated into an expression or command c) Only in the presentation layer (HTML) d) In a utility function without external data	B
	Operational	To verify if the SQL injection vulnerability actually occurs in the application, which test input is most suitable to be entered into the field indicated in the scenario?	a) A long text without special characters b) A simple numerical value c) A value containing an expression/command interpretable by the code d) An empty value	C
	Terminological	After entering an input specially crafted to attempt to exploit SQL injection, what is the most appropriate next step to confirm if the vulnerability is exploitable?	a) Verify if the application continues without compilation errors b) Observe if the result or output was altered by the injected command c) Check only a generic error message d) Restart the server	B
SC2 - XSS	Conceptual	Observing the XSS scenario and the SAST report, which interface element is the most likely candidate to be the vulnerable point?	a) User input data is displayed directly without sanitization b) The problem is solely error handling c) The failure occurs due to system slowness d) The risk lies only in logs	A
	Contextual	Which of the snippets below is most characteristic of a cross-site scripting (XSS) vulnerability as illustrated in the scenario?	a) At the moment the value is read b) On the line where the value is directly inserted into the HTML output c) Only in the database layer d) In a function without external input	B
	Operational	Which of the inputs below is most appropriate to test if the XSS vulnerability in the scenario is exploitable?	a) Plain text without special characters b) Simple number c) Input with an executable script/command (e.g., `<script>`) d) Empty field	C
	Terminological	When executing the test with the XSS payload, which evidence most clearly indicates that the vulnerability was successfully exploited?	a) Verify compilation error b) Observe output alteration c) Verify script execution in the browser (display/behavior) d) Restart server	C

Table 3: Questionnaire structure per scenario, analytical dimension, and scale type

Scenario	Dimension	Question Objective	Scale (1–5)
SC1	Conceptual	When analyzing the SAST report for this scenario, to what extent do you understand what the described SQL injection vulnerability represents in technical terms?	1 = *No understanding* 5 = *Full understanding*
	Contextual	Based on the SAST report, how clear is it to you in which specific part of the code the SQL injection vulnerability occurs?	1 = *Not clear at all* 5 = *Very clear*
	Terminological†	To what extent do the technical terms used in the report (e.g., rule names, severity, vulnerability description) hinder your understanding of this SQL injection issue?	1 = *Not hindering* 5 = *Completely hindering*
	Operational	How confident do you feel in proposing an adequate fix for the presented SQL injection vulnerability using only the information available in the SAST report?	1 = *Not confident* 5 = *Totally confident*
	Effort†	If you were working on a real project, what would be the perceived level of effort to fix this SQL injection vulnerability, considering only the support provided by the SAST report?	1 = *No effort* 5 = *Extremely high*
SC2	Conceptual	When analyzing the SAST report for this scenario, to what extent do you understand what the described cross-site scripting (XSS) vulnerability represents in technical terms?	1 = *No understanding* 5 = *Full understanding*
	Contextual	Based on the SAST report, how clear is it to you in which specific part of the code the XSS vulnerability occurs?	1 = *Not clear at all* 5 = *Very clear*
	Terminological†	To what extent do the technical terms used in the report (e.g., rule names, severity, vulnerability description) hinder your understanding of this XSS issue?	1 = *Not hindering* 5 = *Completely hindering*
	Operational	How confident do you feel in proposing an adequate fix for the presented XSS vulnerability using only the information available in the SAST report?	1 = *Not confident* 5 = *Totally confident*
	Effort†	If you were working on a real project, what would be the perceived level of effort to fix this XSS vulnerability, considering only the support provided by the SAST report?	1 = *No effort* 5 = *Extremely high*
General	Difficulty	Evaluate the participant's perceived difficulty in correctly identifying the vulnerabilities in the two presented scenarios.	1 = *Very easy* 5 = *Very difficult*
	Confidence	Evaluate the participant's level of confidence that their answers reflect a good understanding of the analyzed SAST alerts.	1 = *Not confident* 5 = *Totally confident*

Reverse-coded prior to analysis so that higher values indicate better perceived cognition across all dimensions.

the barriers and participants' cognitive self-perception, their profiles, and their contextualization, terminological, and operational capabilities regarding SAST alerts. All statistical tests adopted a two-sided significance level of $\alpha = 0.05$, corresponding to a 95% confidence level.

The four McNemar tests were treated as independent hypotheses per cognitive dimension, rather than a single family of comparisons. The Holm correction was applied exclusively to the Spearman correlation families (Tables 8 and 9), where multiple coefficients share the same dependent variables.

Discrepancies were resolved by a third researcher. Since the answer keys were derived directly from the OWASP Top 10 guidelines, inter-rater reliability for correct answers does not apply; validation focused on the mapping between the construct and the dimension rather than the correctness of the responses, as the performance questions are objective in nature and have predefined answer keys, as presented in Table 2. The self-reported measures are presented in Table 3.

5 Results

This section presents the results of the analysis of the 30 novice participants who answered the survey.

5.1 Sample Characterization

The final sample consisted of 30 participants, all classified as beginners, with 36.7% having less than one year of experience and 63.3% having between one and three years of experience. Regarding the area of expertise, we have a predominance of students/interns (46.7%), followed by developers (36.7%) and, to a lesser extent, testers/QA professionals (16.7%).

Regarding the frequency of use of SAST tools, most of the participants (96.7%) reported using them sporadically, while only 3.3% indicated that they never use this type of tool.

These results characterize a sample with low practical experience and limited exposure to SAST tools, reflecting a training profile. This characteristic is particularly relevant to this study, as it can directly influence how security reports are interpreted, especially in scenarios that require a deeper conceptual understanding.

5.2 Overall Performance

Participants demonstrated strong performance in conceptual, contextual, operational, and terminological dimensions. However, performance was significantly lower in the conceptual dimension within the SQL Injection scenario. Table 4 presents the participants' success rates by dimension and scenario. Tables 4 and 5 summarize objective performance on the scenario questions listed in Table 2, whereas Table 3 reports the separate self-reported perception measures.

Table 4: Success rate by dimension and scenario

Dimension	SC1 (SQL Injection)	SC2 (XSS Scenario)
Conceptual	13.3% (4/30)	83.3% (25/30)
Contextual	83.3% (25/30)	80.0% (24/30)
Terminological	90.0% (27/30)	80.0% (24/30)
Operational	90.0% (27/30)	73.3% (22/30)

A striking difference is observed in the conceptual dimension between scenarios, with substantially lower performance in SQL Injection (13.3%) compared to the XSS scenario (83.3%). In the other dimensions, participants exhibited high performance in both scenarios, with minor variations.

Participant responses were binary-coded, assigning a value of 1 for correct answers and 0 for incorrect ones. For each analyzed dimension, an aggregate score was calculated, corresponding to the mean performance across the two experimental scenarios (SQL Injection and XSS). Thus, scores ranged from [0,1], where higher values indicate a greater number of correct answers, allowing for a direct comparison between dimensions.

Table 5: Average performance score by dimension (normalized scores)

Dimension	Mean	SD
Terminological	0.850	0.298
Contextual	0.817	0.334
Operational	0.817	0.334
Conceptual	0.483	0.278

Scores computed per participant as the mean of the two experimental scenarios (SC1 and SC2), ranging from 0 to 1. Rows sorted by descending mean. $n = 30$.

Based on the aggregated scores per dimension, high performance was observed in most of the analyzed dimensions. The highest means were identified in the terminological (M = 0.85, SD = 0.30), contextual (M = 0.82, SD = 0.33), and operational (M = 0.82, SD = 0.33) dimensions, while the conceptual dimension presented the lowest average value (M = 0.48, SD = 0.28), reflecting the poor performance observed in the SQL Injection scenario, as shown in Table 4.

It is important to emphasize that the higher conceptual performance stems from the aggregation of the two scenarios, masking the low performance observed in the SQL Injection scenario. This result indicates that while participants demonstrate the ability to understand concepts in less complex scenarios (XSS scenario), they face significant difficulties when the problem requires greater conceptual depth (SQL injection).

On the other hand, the relatively lower performance in the conceptual dimension suggests greater difficulty in the practical application of this understanding, especially in proposing vulnerability analysis and remediation actions.

Key Finding

Although participants demonstrated strong performance in operational, terminological, and contextual dimensions, a notable limitation was observed in the conceptual dimension, likely influenced by the type of vulnerability (e.g., SQL Injection or XSS) and the level of cognitive processing required.

5.3 Scenario Comparison

To investigate differences in participant performance between the evaluated scenarios, a paired comparison was conducted between the SQL Injection (SC1) and XSS Scenario (SC2) conditions for

each analyzed dimension. Considering the binary nature of the data (0 = error; 1 = correct) and the within-subjects design, we applied McNemar's test for paired samples. This test evaluates whether the proportion of disagreements between the two scenarios is asymmetrical, without assuming normality or data continuity. The results of McNemar's test for each dimension are presented in Table 6.

Table 6: Paired scenario comparison — McNemar's Test

Dimension	b	c	χ^2	p	φ	Interpretation
Conceptual	21	0	21.000	<.001	0.837	Significant difference
Contextual	2	3	0.200	.655	0.082	Not significant
Terminological	1	4	1.800	.180	0.245	Not significant
Operational	0	5	5.000	.025	0.408	Significant difference

b = incorrect on SC1, correct on SC2; c = correct on SC1, incorrect on SC2. φ: small ≥.10; medium ≥.30; large ≥.50. McNemar's test without continuity correction; $df = 1$; $n = 30$.

A statistically significant difference was observed in the conceptual dimension ($\chi^2(1) = 21.000, p < .001, \varphi = .837$), highlighting substantially lower performance in the SQL Injection scenario compared to the XSS scenario with a result representing a large effect size. Additionally, a significant difference was identified in the operational dimension ($\chi^2(1) = 5.000, p = .025, \varphi = .408$), suggesting greater difficulty in performing practical actions within the SQL Injection scenario, with a medium-magnitude effect. Conversely, no statistically significant differences were observed in the contextual ($\chi^2(1) = 0.200, p = .655$) and terminological ($\chi^2(1) = 1.800, p = .180$) dimensions, indicating that the ability to locate the vulnerability in the code and interpret technical terms remains relatively stable across scenarios.

These results suggest that the impact of the vulnerability type on participant performance depends on the cognitive dimension being analyzed, primarily affecting conceptual understanding and, to a lesser extent, operational execution.

Key Finding

The impact of the vulnerability type on performance is not uniform; it is significantly more pronounced in the conceptual dimension and, to a lesser degree, in the operational dimension.

5.4 Perceived Cognition

In addition to objective performance, we evaluted the participants' perceived cognition regarding the evaluated scenarios. This analysis aimed to investigate how participants perceive their own understanding, clarity, operational capability, and effort when interpreting SAST reports.

For this purpose, we applied Likert-type scales separately to each scenario (SQL Injection and XSS), covering the conceptual, contextual, terminological, and operational dimensions, as well as perceived effort. Items were analyzed independently by dimension, with descriptive statistics (mean (M) and standard deviation (SD)) reported for each scenario. To ensure interpretive consistency, negatively oriented items (terminology and effort) were reverse-coded so that higher values represented a better cognitive perception.

The descriptive statistics for perceived cognition in each scenario are presented in Table 7.

Overall, the results indicate moderate levels of perceived cognition in both scenarios. The observed means ranged between 2.23 and 3.40, suggesting that participants do not perceive themselves as fully confident or entirely capable when interpreting SAST reports.

In the SQL Injection scenario (SC1), the highest mean was observed in the contextual dimension (M = 3.17), indicating a relatively higher perception of identifying the vulnerability's location in the code. In contrast, the lowest mean was identified in the operational dimension (M = 2.50), suggesting difficulties in proposing practical actions.

In the XSS scenario (SC2), the values were generally slightly lower, with the operational dimension (M = 2.23) presenting the lowest value among all analyzed measures.

Furthermore, it is observed that perceived effort remained relatively high and consistent across scenarios (SC1: M = 3.37; SC2: M = 3.40), indicating that participants perceive the tasks as demanding, regardless of the type of vulnerability analyzed.

Key Finding

Despite high performance across several dimensions, participants report only moderate levels of understanding and high perceived effort, indicating low confidence in interpreting SAST reports.

5.5 Relationships Between Performance and Self-Assessment

To investigate the relationships between objective performance and self-assessment variables, we performed a correlation analysis using the Spearman coefficient (ρ), considering the non-parametric nature of the data. Associations were analyzed between performance scores, i.e., regarding the correct answers across both scenarios in the conceptual, contextual, operational, and terminological dimensions, and the self-assessment measures of difficulty and confidence. Table 8 presents the correlation results.

Overall, the correlations between performance dimensions and self-assessment variables were weak and non-significant, indicating an absence of consistent association between objective performance and participants' perception.

Among the performance dimensions, no significant associations were observed after the Holm correction, including the correlation between operational and terminological performance ($\rho = 0.413$, $p_{\text{raw}} = .023$, $p_{\text{Holm}} = .327$), which did not survive the control for multiple comparisons.

Regarding the self-assessment variables, a significant negative correlation between perceived difficulty and confidence stood out ($\rho = -0.569$, $p_{\text{raw}} = .001$, $p_{\text{Holm}} = .016$), indicating that participants who perceive greater difficulty tend to report lower confidence.

However, no significant correlations were observed between performance and the measures of difficulty or confidence, suggesting that participants' self-assessment is not directly associated with their objective performance. The p-values for all non-significant coefficients are reported in Table 8, confirming the lack of consistent association between objective performance and the global perception of the participants.

Table 7: Perceived cognition by dimension and scenario — frequency distribution and descriptive statistics

Dimension \ Scenario	SC1							SC2						
	1 *n*(%)	2 *n*(%)	3 *n*(%)	4 *n*(%)	5 *n*(%)	M	SD	1 *n*(%)	2 *n*(%)	3 *n*(%)	4 *n*(%)	5 *n*(%)	M	SD
Conceptual[a]	2 (6.7)	11 (36.7)	12 (40.0)	4 (13.3)	1 (3.3)	2.70	0.92	3 (10.0)	15 (50.0)	6 (20.0)	5 (16.7)	1 (3.3)	2.53	1.01
Contextual[b]	0 (0.0)	7 (23.3)	11 (36.7)	12 (40.0)	0 (0.0)	3.17	0.79	3 (10.0)	9 (30.0)	12 (40.0)	5 (16.7)	1 (3.3)	2.73	0.98
Terminological[c†]	6 (20.0)	5 (16.7)	14 (46.7)	4 (13.3)	1 (3.3)	2.63	1.07	3 (10.0)	9 (30.0)	14 (46.7)	3 (10.0)	1 (3.3)	2.67	0.92
Operational[d]	5 (16.7)	9 (30.0)	12 (40.0)	4 (13.3)	0 (0.0)	2.50	0.94	7 (23.3)	11 (36.7)	10 (33.3)	2 (6.7)	0 (0.0)	2.23	0.90
Effort[e†]	0 (0.0)	3 (10.0)	15 (50.0)	10 (33.3)	2 (6.7)	3.37	0.76	0 (0.0)	6 (20.0)	9 (30.0)	12 (40.0)	3 (10.0)	3.40	0.93

Table 8: Spearman correlation matrix: performance dimensions and self-assessment measures (Holm-corrected)

	(1) Conceptual	(2) Contextual	(3) Operational	(4) Terminological	(5) Difficulty	(6) Confidence
(1) Conceptual	–					
(2) Contextual	−0.224 (.235/1.000)	–				
(3) Operational	0.135 (.477/1.000)	0.264 (.159/1.000)	–			
(4) Terminological	0.000 (1.000/1.000)	0.320 (.085/1.000)	0.413 (.023/.327)	–		
(5) Difficulty	0.028 (.882/1.000)	−0.053 (.783/1.000)	−0.119 (.531/1.000)	−0.169 (.371/1.000)	–	
(6) Confidence	0.103 (.590/1.000)	−0.050 (.792/1.000)	0.177 (.348/1.000)	−0.118 (.535/1.000)	−0.569* (.001/.016)	–

Values reported as ρ ($p_{\text{uncorrected}}/p_{\text{Holm}}$). $n = 30$; 15 pairwise comparisons; $\alpha = .05$. * significant after Holm–Bonferroni sequential correction [11]. After correction, only Difficulty × Confidence remains significant ($\rho = -0.569$, $p_{\text{Holm}} = .016$).

Key Finding

No consistent associations were observed between objective performance and self-assessment measures (difficulty and confidence), indicating that participants do not calibrate their global perceptions to their actual performance.

5.6 Relationship Between Performance and Perception

Complementing the global self-assessment measures, we examined the alignment between objective performance and perceived cognition by dimension. We computed Spearman correlations (ρ) between actual scores and perceived dimensions, analyzing each scenario separately (SQL Injection—SC1 and XSS—SC2). The results are presented in Table 9.

Table 9: Spearman correlation between objective performance and perceived cognition by scenario (Holm-corrected)

Dimension	SC1 (SQL Injection)		SC2 (XSS)	
	ρ	p / p_{Holm}	ρ	p / p_{Holm}
Conceptual	−0.518*	.003 / .020	0.139	.463 / 1.000
Contextual	0.210	.266 / 1.000	0.172	.363 / 1.000
Operational	−0.061	.750 / 1.000	0.142	.454 / 1.000

* significant after Holm correction. $n = 30$; 6 comparisons; $\alpha = .05$. Holm–Bonferroni sequential correction applied across all 6 pairwise comparisons [11]. The negative correlation in Conceptual (SC1) remains significant after correction ($\rho = -0.518$, $p_{\text{Holm}} = .020$). This pattern diverges from the Dunning–Kruger effect [15] and is more consistent with conservative calibration under uncertainty [14]: the four participants who answered correctly reported lower perceived understanding (scores 1–2), suggesting underestimation of competence rather than overestimation.

In the SQL Injection scenario (SC1), a significant negative correlation was observed in the conceptual dimension ($\rho = -0.518$, $p_{\text{raw}} = .003$, $p_{\text{Holm}} = .020$), indicating that participants with better actual performance tended to report lower perceived understanding. This pattern diverges from the Dunning-Kruger effect [15], which postulates that low-performing individuals overestimate their competence because they lack the metacognitive ability to recognize the quality of their own performance. The observed finding is the opposite: the four participants who answered correctly assessed their understanding as low (perception 1–2), while the 26 who failed reported moderate to high perception (perception 2–5). This pattern is more consistent with conservative calibration under uncertainty [14]: participants who succeeded might have done so without full confidence in their own reasoning, leading to an underestimation of actual competence. Given the small number of correct answers in SC1 ($n = 4$), this interpretation is exploratory and should be replicated with larger samples.

Overall, the findings indicate a dissociation between objective performance and perceived cognition, most evident in the SQL Injection scenario. This pattern suggests that participants' perception does not consistently track their actual performance, especially in tasks requiring greater conceptual understanding.

Key Finding

A systematic misalignment is observed between objective performance and perceived cognition, particularly in the SQL Injection scenario, where higher performance is not associated with greater perceived understanding. These findings suggests limitations in participants' cognitive ability, indicating difficulty in accurately assessing their own competence when performing SAST report analysis tasks.

Integrated, the results indicate that both perceived cognition and self-assessment measures do not consistently reflect objective performance, reinforcing the presence of a cognitive misalignment in novices when interpreting SAST reports.

6 Discussion

The results of this study reveal a consistent pattern of dissociation between objective performance and participant perception when interpreting SAST reports, especially in scenarios that require a greater conceptual understanding.

Initially, we observed that the participants demonstrated high performance in the contextual, terminological, and operational dimensions, but significantly lower performance in the conceptual dimension within the SQL Injection scenario. This result suggests that, while participants are capable of locating vulnerabilities in the code, interpreting technical terms, and performing operational actions, they face difficulties in achieving a deeper understanding of the security problem.

This finding is reinforced by the comparison between scenarios, which evidenced that the impact of vulnerability is not uniform across cognitive dimensions. The significant difference in conceptual dimension indicates that more complex vulnerabilities impose a higher intrinsic cognitive load, hindering the construction of an adequate mental model of the problem. This result aligns with studies indicating that tasks with high conceptual complexity demand greater cognitive effort and directly affect an individual's capacity for action [6, 16, 23], indicating that tasks with high conceptual complexity demand greater cognitive effort and directly affect an individual's capacity for action.

Regarding perceived cognition, participants reported only moderate levels of understanding and high perceived effort, regardless of objective performance. This pattern suggests that participants recognize the complexity of the task, but do not necessarily succeed in translating this perception into an accurate assessment of their own performance.

The analysis of the relationships between performance and perceived cognition revealed the primary finding of this study: the presence of a misalignment between objective performance and perception. In the SQL Injection scenario, a significant negative correlation was observed in the conceptual dimension, indicating that participants with better performance tended to underestimate their own understanding. This pattern, contrary to the one predicted by the Dunning-Kruger effect [15], suggests conservative calibration: in tasks of high conceptual complexity and low familiarity, even those who succeed may not recognize the soundness of their own reasoning, resulting in a perception of competence below actual performance [14]. This dissociation has relevant practical implications: novices who underestimate their understanding may seek unnecessary external validation or hesitate in decision-making, even when their analysis is technically adequate.

Additionally, the lack of correlation between performance and global self-assessment measures (difficulty and confidence) reinforces this pattern, indicating that both specific perception and the general evaluation of participants do not consistently reflect their actual performance. However, the strong relationship between perceived difficulty and confidence indicates that these dimensions are internally consistent, albeit disconnected from objective performance.

These results can be interpreted in light of Cognitive Load Theory [26], which suggests that tasks with high intrinsic complexity can compromise an individual's ability to monitor their own learning and performance. In the context of SAST reports, the combination of technical terminology, lack of context, and the need for inference regarding system behavior can generate cognitive overload, hindering both understanding and self-assessment.

From a practical standpoint, the findings of this study have relevant implications for the use of SAST tools in real-world environments. The lack of alignment between performance and perception suggests that novices can make inadequate decisions when interpreting reports, either by underestimating or overestimating their understanding. This can lead to both overlooked vulnerabilities and incorrect execution of remediation actions.

In this sense, the results indicate the need for the development of cognitive support mechanisms in SAST tools, such as clearer explanations, contextualization of vulnerabilities, and integration with practical examples. Furthermore, approaches based on Large Language Models (LLMs) can act as a support to reduce cognitive load and assist in the interpretation of reports.

Finally, from a scientific standpoint, this study contributes by empirically evidencing the presence of cognitive misalignment in software security tasks, expanding the understanding of the barriers faced by novices. Unlike previous studies that focus predominantly on the technical aspects of tools, this study highlights the importance of cognitive and perceptual factors in the effectiveness of SAST use.

Answering the Research Question

RQ: What are the primary cognitive and technical barriers that novices face when interpreting SAST tool reports? The primary barriers are concentrated in the conceptual dimension, especially in higher-complexity scenarios such as SQL Injection: although participants can interpret technical terms, locate vulnerabilities, and execute operational actions, they struggle to deeply understand the vulnerabilities' mechanics and impact. A misalignment between objective performance and perceived cognition was also identified, evidencing metacognitive limitations that may compromise decision-making during analysis and remediation. These barriers show that SAST challenges are technical and cognitive alike, reinforcing the need for approaches that integrate both forms of support.

7 Threats to Validity

This section presents the main threats to the validity of the study, organized into categories Wohlin et al. [30], namely the validity of the construct, internal, external and conclusion.

Construct Validity. It refers to the adequacy of the variables used to represent the investigated concepts [30]. In this study, performance was operationalized using objective questions categorized into conceptual, contextual, terminological, and operational dimensions. Although this structure was grounded in the technical literature and the objectives of the study, there is a risk that these dimensions may not fully capture the complexity of the cognitive processes involved in the interpretation of SAST reports. Furthermore, perceived cognition was measured using Likert scales, which rely on the subjective interpretation of the participants. Although the items varied between positive and negative aspects, the responses

for the negative items were reverse-coded to allow for standardization during the statistical analysis; nonetheless, responses may have been influenced by individual biases, such as central tendency or social desirability. Additionally, the operationalization of "novice" based solely on years of experience (3 years) and self-declared frequency of SAST usage represents a limitation to the construct validity. The duration of experience does not directly measure proficiency levels. Future studies should complement this criterion with an objective security knowledge pre-test to ensure participant homogeneity and strengthen construct validity. To mitigate these limitations, the questions were developed based on real-world scenarios derived from the OWASP Top 10 and validated by experts, seeking to ensure alignment between the theoretical constructs and the measures used.

Internal Validity. It concerns the existence of uncontrolled factors that might influence the observed results [30]. Because this study was conducted through an online survey, there was no control over the execution environment, which could lead to interferences such as distractions, external material consultation, or varying levels of participant attention. Additionally, the interpretation of the scenarios may have been influenced by the participants' prior knowledge of specific vulnerabilities, which could affect performance independently of the clarity of the presented report. As a mitigation strategy, standardized scenarios were used and presented uniformly to all participants, alongside the application of inclusion criteria to restrict the sample to individuals with a similar profile (novices). The possibility of the questions being poorly formulated or lacking clarity was taken into consideration during the study. To mitigate these risks, a review was conducted with an experienced researcher in the software testing field, and a pilot study was carried out with participants representing the target sample.

External Validity. It refers to the generalizability of the findings [30]. The sample was composed exclusively of novice participants with up to three years of experience and limited use of SAST tools. Consequently, the results may not be generalizable to more experienced professionals or more complex organizational contexts. Furthermore, the scenarios were based on specific vulnerabilities (SQL Injection and XSS), which may limit the generalization of the findings to other types of vulnerabilities or different SAST tools. However, the use of scenarios based on the OWASP Juice Shop platform, analyzed with widely adopted tools such as SonarQube, enhances the real-world representativeness of the experimental environment. This approach ensures that the study reflects conditions closely aligned with practical applications, the Juice Shop provides a simulation of a realistic web system with representative vulnerabilities, while SonarQube mirrors static analysis tools commonly employed in industry, thereby increasing the practical relevance and applicability of the findings.

Conclusion Validity. It refers to the suitability of the statistical techniques used and the interpretation of the results [30]. Considering the small sample size and the nature of the data, appropriate tests were used for each variable type: McNemar's test for paired comparisons with binary data and Spearman's correlation for associations between ordinal variables, thereby avoiding assumptions of normality. Nevertheless, the limited number of participants may reduce the statistical power of the analyzes, especially when detecting more subtle associations. Additionally, the presence of binary data may influence the interpretation of effect size measures, such as correlation coefficients. To mitigate these limitations, the results were interpreted with caution, prioritizing consistent patterns across analyzes and avoiding inferences beyond the observed statistical support.

8 Conclusion

This study investigates the cognitive barriers faced by novice practitioners when interpreting SAST reports, considering various dimensions of performance and perception within SQL Injection and XSS vulnerability scenarios.

The results demonstrated that while participants exhibit strong performance in localization, terminological interpretation, and execution tasks, there is a significant limitation in conceptual understanding, particularly in more complex scenarios. Furthermore, a misalignment was identified between objective performance and perceived cognition, indicating that participants do not accurately assess their own level of understanding.

From a scientific point of view, this work contributes by empirically demonstrating the presence of cognitive misalignment in software security tasks, expanding the understanding of the human factors involved in the interpretation of vulnerabilities. From a practical perspective, the findings suggest the need to enhance SAST tools by focusing on cognitive support, explainability, and improved contextualization of the presented vulnerabilities.

For future work, we suggest expanding the sample to include participants with varying levels of experience, incorporating additional types of vulnerabilities, and investigating LLM-based support approaches to reduce cognitive load and improve the interpretation of security reports. We also propose developing and validating a scale to assess novices' self-perceived competence in analyzing and operationalizing SAST reports. This instrument could be structured around the dimensions identified in this study (conceptual, contextual, terminological, and operational), enabling a more precise assessment of the misalignment between objective performance and perception, as well as supporting educational interventions and improvements in security tools.

Ethical Considerations on the Use of AI

We used Large Language Models for language refinement, but all analysis and interpretation remain our sole responsibility.

Artifact Availability

We provide the following resources: a survey, anonymous data, the statistical analysis software package, and the runtime environment at the following link: https://github.com/ASTRID-RESEARCH/2026-SBES-Package-Replication.git

Acknowledgments

This work was developed with support from the National Council for Scientific and Technological Development - CNPq (Process 312173/2025-3) and Brazilian Federal Agency for Support and Evaluation of Graduate Education - (CAPES) with Financing Code 001.

References

[1] Amit Seal Ami, Kevin Moran, Denys Poshyvanyk, and Adwait Nadkarni. 2024. "False negative - that one is going to kill you": Understanding Industry Perspectives of Static Analysis based Security Testing. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 3979–3997. doi:10.1109/SP54263.2024.00019

[2] Vairam Arunachalam and William Sasso. 1996. Cognitive processes in program comprehension: An empirical analysis in the Context of software reengineering. *Journal of Systems and Software* 34 (9 1996), 177–189. Issue 3. doi:10.1016/0164-1212(95)00074-7

[3] Albert Bandura et al. 1986. Social foundations of thought and action. *Englewood Cliffs, NJ* 1986, 23-28 (1986), 2.

[4] Bruno Barroso, Leonardo da Silva, and Rafael de Mello. 2024. Caracterizando Condutores de Carga Cognitiva na Prática de Testes Unitários. In *Anais do IX Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES 2024)*. Sociedade Brasileira de Computação - SBC, 70–81. doi:10.5753/washes.2024.2676

[5] Gareth Bennett, Tracy Hall, Steve Counsell, Emily Winter, and Thomas Shippey. 2024. Do Developers Use Static Application Security Testing (SAST) Tools Straight Out of the Box? A large-scale Empirical Study. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (Barcelona, Spain) *(ESEM '24)*. Association for Computing Machinery, New York, NY, USA, 454–460. doi:10.1145/3674805.3690750

[6] Ouhao Chen, Fred Paas, and John Sweller. 2023. A Cognitive Load Theory Approach to Defining and Measuring Task Complexity Through Element Interactivity. *Educational Psychology Review* 35 (6 2023), 63. Issue 2. doi:10.1007/s10648-023-09782-w

[7] Lisa Nguyen Quang Do, James R. Wright, and Karim Ali. 2022. Why Do Software Developers Use Static Analysis Tools? A User-Centered Study of Developer Needs and Motivations. *IEEE Transactions on Software Engineering* 48 (3 2022), 835–847. Issue 3. doi:10.1109/TSE.2020.3004525

[8] Fabian Fagerholm, Michael Felderer, Davide Fucci, Michael Unterkalmsteiner, Bogdan Marculescu, Markus Martini, Lars Göran Wallgren Tengberg, Robert Feldt, Bettina Lehtelä, Balázs Nagyváradi, and Jehan Khattak. 2022. Cognition in Software Engineering: A Taxonomy and Survey of a Half-Century of Research. *Comput. Surveys* 54 (1 2022), 1–36. Issue 11s. doi:10.1145/3508359

[9] Clarisse Feio, Nuno Santos, Nelson Escravana, and Bernardo Pacheco. 2024. An Empirical Study of DevSecOps Focused on Continuous Security Testing. In *2024 IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*. 610–617. doi:10.1109/EuroSPW61312.2024.00074

[10] Thomas Fritz, Andrew Begel, Sebastian C. Müller, Serap Yigit-Elliott, and Manuela Züger. 2014. Using psycho-physiological measures to assess task difficulty in software development. In *Proceedings of the 36th International Conference on Software Engineering* (New York, NY, USA). ACM, 402–413. doi:10.1145/2568225.2568266

[11] Sture Holm. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70.

[12] Brittany Johnson, Yoonki Song, Emerson Murphy-Hill, and Robert Bowdidge. 2013. Why don't software developers use static analysis tools to find bugs?. In *2013 35th International Conference on Software Engineering (ICSE)*. 672–681. doi:10.1109/ICSE.2013.6606613

[13] Mark Kasunic. 2005. *Designing an Effective Survey* (1 ed.). Vol. 1. Carnegie Mellon. 1–143 pages.

[14] Asher Koriat. 2012. The self-consistency model of subjective confidence. *Psychological Review* 119 (1 2012), 80–113. Issue 1. doi:10.1037/a0025648

[15] Justin Kruger and David Dunning. 1999. Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments. *Journal of Personality and Social Psychology* 77 (1999), 1121–1134. Issue 6. doi:10.1037/0022-3514.77.6.1121

[16] Erik Lintunen, Viljami Salmela, Petri Jarre, Tuukka Heikkinen, Markku Kilpeläinen, Markus Jokela, and Antti Oulasvirta. 2024. Cognitive abilities predict performance in everyday computer tasks. *International Journal of Human-Computer Studies* 192 (12 2024), 103354. doi:10.1016/j.ijhcs.2024.103354

[17] Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. 2022. Metacognition and Self-Regulation in Programming Education: Theories and Exemplars of Use. *ACM Transactions on Computing Education* 22 (12 2022), 1–31. Issue 4. doi:10.1145/3487050

[18] Jefferson Seide Molléri, Kai Petersen, and Emilia Mendes. 2016. Survey Guidelines in Software Engineering. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (New York, NY, USA). ACM, 1–6. doi:10.1145/2961111.2962619

[19] Marcus Nachtigall, Michael Schlichtig, and Eric Bodden. 2022. A large-scale study of usability criteria addressed by static analysis tools. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis* (New York, NY, USA). ACM, 532–543. doi:10.1145/3533767.3534374

[20] National Institute of Standards and Technology (NIST). 2026. National Vulnerability Database: Dashboard. https://nvd.nist.gov/general/nvd-dashboard. Acessado em: 27 de abril de 2026.

[21] OWASP Foundation. 2025. *A05:2025 Injection*. Open Web Application Security Project (OWASP). https://owasp.org/Top10/2025/A05_2025-Injection/ Acesso em: 16 fev. 2026.

[22] Roxana Quintero-Manes and Camilo Vieira. 2025. Differentiated measurement of cognitive loads in computer programming. *Journal of Computing in Higher Education* 37 (9 2025), 945–962. Issue 3. doi:10.1007/s12528-024-09411-7

[23] Dasha A. Sandra and A. Ross Otto. 2018. Cognitive capacity limitations and Need for Cognition differentially predict reward-induced cognitive effort expenditure. *Cognition* 172 (3 2018), 101–106. doi:10.1016/j.cognition.2017.12.004

[24] Hugo H. Schoonewille, Werner Heijstek, Michel R.V. Chaudron, and Thomas Kühne. 2011. A cognitive perspective on developer comprehension of software design documentation. In *Proceedings of the 29th ACM international conference on Design of communication* (New York, NY, USA). ACM, 211–218. doi:10.1145/2038476.2038517

[25] John Sweller. 1988. Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science* 12, 2 (April 1988), 257–285. doi:10.1207/s15516709cog1202_4

[26] John Sweller. 1988. Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science* 12 (4 1988), 257–285. Issue 2. doi:10.1207/s15516709cog1202_4

[27] Mohammad Tahaei, Kami Vaniea, Konstantin (Kosta) Beznosov, and Maria K Wolters. 2021. Security Notifications in Static Analysis Tools: Developers' Attitudes, Comprehension, and Ability to Act on Them. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA). ACM, 1–17. doi:10.1145/3411764.3445616

[28] Laura Toma and Jan Vahrenhold. 2018. Self-Efficacy, Cognitive Load, and Emotional Reactions in Collaborative Algorithms Labs - A Case Study. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (New York, NY, USA). ACM, 1–10. doi:10.1145/3230977.3230980

[29] Zachary Douglas Wadhams, Clemente Izurieta, and Ann Marie Reinhold. 2024. Barriers to Using Static Application Security Testing (SAST) Tools: A Literature Review. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering Workshops* (New York, NY, USA). ACM, 161–166. doi:10.1145/3691621.3694947

[30] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. 2012. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.

[31] Xueqi Yang, Zhe Yu, Junjie Wang, and Tim Menzies. 2021. Understanding static code warnings: An incremental AI approach. *Expert Systems with Applications* 167 (4 2021), 114134. doi:10.1016/j.eswa.2020.114134

[32] Yunze Zhao, Wentao Guo, Harrison Goldstein, Daniel Votipka, Kelsey R. Fulton, and Michelle L. Mazurek. 2025. A Qualitative Analysis of Fuzzer Usability and Challenges. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security* (New York, NY, USA). ACM, 2504–2518. doi:10.1145/3719027.3765055

[33] Ozan Özkan, Önder Babur, and Mark van den Brand. 2023. Refactoring with domain-driven design in an industrial context. *Empirical Software Engineering* 28 (7 2023), 94. Issue 4. doi:10.1007/s10664-023-10310-1