Applying DevSecOps Approach in Legacy Computing Infrastructures: A Case Study in Public Sector of Brazil

João C. C. Lima[1], Francisco R. M. Campos[1], Rafael L. Gomes[2], Emanuel B. Rodrigues[1], Rossana M. C. Andrade[1], Clenival L. Silva[3], Daniel C. Bentes[3], and Alexandre S. Cialdini[3]

[1] Federal University of Ceará (UFC), Fortaleza, Ceará, Brazil.
{jcarloslima,ramon}@alu.ufc.br, {emanuel,rossana}@dc.ufc.br
[2] State University of Ceará (UECE), Fortaleza, Ceará, Brazil
rafa.lopes@uece.br
[3] State Secretariat of Planning and Management of Ceará (SEPLAG), Fortaleza, Ceará, Brazil.
{clenival.lopes,daniel.bentes,alexandre.cialdini}@seplag.ce.gov.br

Abstract. Legacy computing environments in the public sector present significant cybersecurity challenges due to outdated systems, technological heterogeneity and complex operational demands. In this context, this paper presents an initial study within a real-world case at a Brazilian governmental institution that applies the DevSecOps methodology in the CI/CD pipeline with two distinct security tools: Static Application Security Testing (SAST) and Vulnerability Management (VM). SAST was applied to assess application security at code level, while VM targeted infrastructure-level risks using metrics such as CVSS and EPSS. The results demonstrate the value of each approach in improving risk visibility and mitigation and the possibility of integrating both tools into a unified DevSecOps workflow, aiming for continuous security and greater operational resilience. This work provides practical insights for public institutions seeking to modernize their cybersecurity posture while addressing the constraints inherent to legacy systems, in alignment with frameworks such as NIST CSF and CIS Controls.

Keywords: Cybersecurity · DevSecOps · CI/CD Pipeline · Application Security Testing · Vulnerability Management · Risk Mitigation.

1 Introduction

The landscape of cyber threats in Brazil is rapidly evolving, characterized by increasingly sophisticated tactics that target individuals and institutions alike. Data from Serasa Experian's 2025 Digital Identity and Fraud Report reveals a stark reality: 51% of Brazilians fell victim to fraud in the last year, with a significant portion (54%) experiencing financial losses [29]. This pervasive insecurity underscores the critical need for robust cybersecurity measures, particularly within government institutions.

Concurrently, the push for digital transformation in public services has led government agencies to develop numerous software solutions. This rapid growth has created highly heterogeneous technological environments, involving a wide range of libraries, platforms, and frameworks [2, 13]. While this fosters innovation, it also increases maintenance complexity, complicates system integration, and heightens information security risks such as update failures and vulnerability exposure [31]. These risks are particularly critical in public institutions like the Secretariat of Planning and Management of Ceará (SEPLAG)[4], which handles sensitive data including personal, fiscal, judicial, and health information. The rising incidence of cyberattacks globally especially in Latin America and Brazil further intensifies the threat landscape [22]. In this context, mitigating vulnerabilities, ensuring service continuity, and complying with data protection regulations such as the LGPD are strategic imperatives [21].

In this complex environment, adopting a DevSecOps methodology becomes essential, as it integrates security practices throughout the Software Development Life Cycle (SDLC), promoting a proactive “security-by-design” approach. This continuous security integration is crucial for managing vulnerabilities in diverse and dynamic public sector infrastructures, aligning with internationally recognized best practices such as the National Institute of Standards and Technology (NIST) Cybersecurity Framework (CSF) [16], which emphasizes continuous risk assessment, protection, and response, and the Center for Internet Security Critical Security Controls (CIS Controls) [3], which provide a prioritized set of defensive measures against common attack vectors.

Within the DevSecOps framework, the Static Application Security Testing (SAST) technique is a integral component of a robust Continuous Integration/Continuous Delivery (CI/CD) pipeline. Since it enables the identification of vulnerabilities directly within the source code [4], SAST allows for early detection of security flaws, adhering to the “shift-left” principle of DevSecOps. This proactive approach is particularly valuable in the early stages of development, minimizing the cost and effort associated with fixing vulnerabilities later in the cycle.

Additionaly, Vulnerability Management (VM) plays a fundamental strategic role, especially given the aforementioned complexity and heterogeneity of public agency computing environments [27]. Unlike homogeneous setups, public institutions contend with a diverse ecosystem of legacy systems, modern applications, varied hardware, and a wide spectrum of user profiles. This process allows for the systematic identification of technical flaws and configuration gaps that could be exploited by malicious actors, safeguarding sensitive data confidentiality, critical system integrity, and public service continuity [7].

Within this context, this paper presents a case study on the application of critical security practices of DevSecOps (in particular SAST and VM) in the context of the Secretariat of Planning and Management of Ceará (SEPLAG) in Brazil. These practices were applied in the IT processes, allowing the evaluation

[4] https://www.seplag.ce.gov.br

of their effectiveness in identifying and mitigating security risks within legacy public sector environments.

The remainder of this paper is organized as follows. Section 2 presents some existing work about DevSecOps and security solutions. Section 3 describes the methodology applied, while Section 4 outlines the experimental setup and discusses the results. Finally, Sections 5 and 6 present the final discussion, conclusions and perspectives of future work.

2 Related Work

This section presents some work on the combined use of SAST, VM and other security tools in DevSecOps pipelines, emphasizing how this combined strategy can strengthen the cybersecurity posture of digital public services.

Riaz et al. [26] analyze the integration of VM, SAST and DAST within the DevSecOps framework as essential strategies to strengthen software security throughout the development lifecycle. By embedding security practices into every phase of DevOps, the study emphasizes the importance of identifying, assessing, and remediating vulnerabilities proactively to prevent security breaches and ensure compliance with industry standards. Through a Multivocal Literature Review (MLR), the authors analyze both academic and industry sources to highlight key metrics for evaluating the effectiveness of DevSecOps implementations. The paper ultimately reinforces the critical role of structured VM and automated testing tools like SAST and DAST in building secure, resilient systems especially in environments where rapid, automated deployments are the norm.

Marandi et al. [14] present a practical approach to integrating VM with SAST and DAST within DevSecOps pipelines, specifically in the context of containerized applications and CI/CD environments. Recognizing the increased security risks introduced by containerization and automated deployments, the research focuses on automated security scanning of software images deployed in cloud infrastructures. It proposes a method using Snyk (for SAST) and StackHawk (for DAST) to scan application images during the build process, providing a dashboard for tracking vulnerabilities and automating fixes. The tools are integrated with GitHub, enabling continuous vulnerability detection and remediation as part of the CI/CD workflow. The study demonstrates that this integration improves security by reducing the time required to identify and address vulnerabilities, thereby strengthening the overall security posture of applications in DevSecOps environments.

A relevant study conducted in a Mexican government organization [5] presents the implementation of a DevSecOps strategy within a responsive infrastructure designed to ensure high availability, cybersecurity, and risk management. The initiative aimed to support the automation of critical processes by leveraging modern technologies such as microservices and container-based architectures, allowing for scalable and fault-tolerant systems. The authors emphasize the use of DevSecOps as an integral solution to reduce project delivery times while enhancing the quality and security of public digital services. Rooted in agile and lean principles, the DevSecOps approach in this case promotes strong collaboration between IT professionals and developers, enabling continuous and secure deployment in complex public-sector environments. This study demonstrates how DevSecOps can be effectively adapted to the specific needs of government data centers to improve both operational efficiency and service delivery to citizens.

Efendi et al. [6] explore the integration of the DevSecOps approach within a public-sector organization, specifically the Public Company Logistic Agency (PCLA), to address challenges in application development such as project delays, frequent late-stage changes, and high vulnerability levels. Through a Systematic Literature Review (SLR) and a mixed-method research approach, the authors identified DevSecOps transformation phases and best practices from various case studies. The study highlights how DevSecOps can help public institutions like PCLA reduce operational costs, enhance software quality, and strengthen security throughout the software development lifecycle. By adopting DevSecOps, PCLA aims to modernize its development processes and align them with industry standards, offering valuable insights for both academics and practitioners in the digital transformation of state-owned enterprises.

As presented by [11], the NIST CSF was applied to assess the cybersecurity posture of a local government organization in Western Australia. Their methodology enabled the quantification of risk across the CSF's core functions and categories, helping to identify specific gaps in people, processes, and technologies. Based on this assessment, strategic recommendations were provided to guide mitigation efforts and strengthen future security capabilities. The authors emphasize the comparative advantages of the NIST CSF over other frameworks, particularly in facilitating structured evaluations. They also note that, despite some implementation challenges, the framework is effective in guiding organizations toward measurable improvements. Future work aims to enhance the assessment tool used in the study, making it more adaptable and accessible for broader adoption.

The main contribution of the present study, in comparison to the cited works, lies in the practical and structured application of security techniques SAST and VM within the legacy infrastructure of a real public-sector organization (SEPLAG), a context still underexplored in the literature. Unlike prior studies that focus on generic DevSecOps strategies or implementations in cloud-native and highly automated environments, this work demonstrates the technical feasibility of adopting DevSecOps practices in traditional legacy systems by integrating tools such as Trivy, Semgrep, and OpenVAS into an existing CI/CD pipeline. It also provides concrete performance and risk metrics (e.g., execution times, CVSS severities, and EPSS scores), while adopting a prioritization approach aligned with the NIST CSF and CIS Controls. By combining empirical evidence with internationally recognized best practices, this study offers a replicable model for public institutions seeking to modernize legacy systems with a strong cybersecurity posture.

3 Methodology

Figure 1 presents the DevSecOps methodology adopted in this work for incorporating security practices into the software development process at SEPLAG, with a focus on two distinct approaches: SAST and VM.

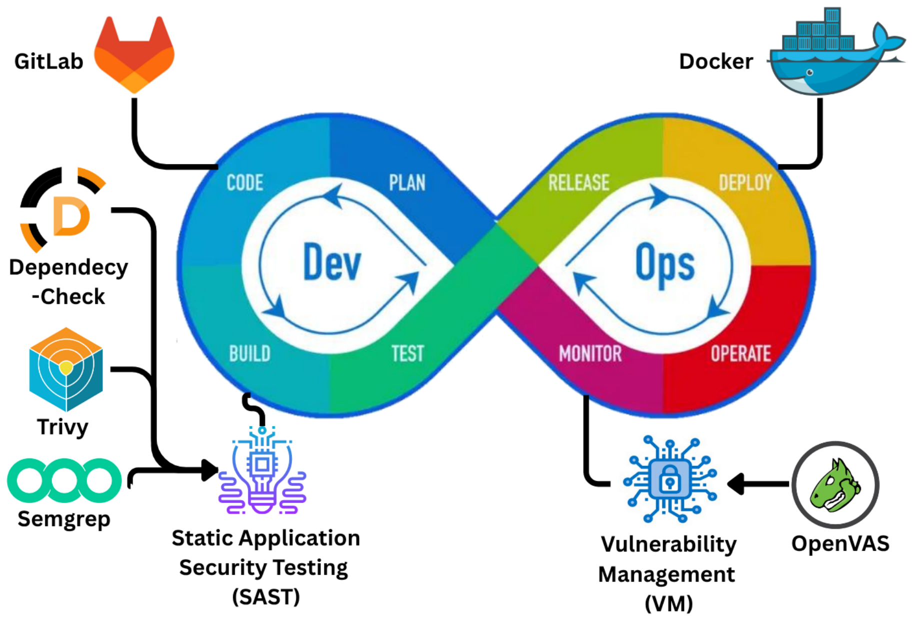

Fig. 1. DevSecOps methodology and security tools used in the case study.

The DevSecOps loop integrates security practices throughout all stages of the software development life cycle, ensuring that security is a continuous and automated concern. During the Plan phase, teams can incorporate threat modeling to anticipate potential risks and define mitigation strategies early. In the Code phase, security starts with developer awareness and secure coding practices, complemented by tools such as Semgrep for SAST. The Build phase benefits from dependency checks (e.g., using tools like Trivy or Dependency-Check) to detect known vulnerabilities in third-party libraries. In the Test phase, automated security testing, including SAST, ensures that vulnerabilities are caught before deployment. The Release and Deploy phases involve secure configuration practices and container scanning (e.g., with Trivy) to validate the runtime environment. Once in Operate and Monitor, VM practices are applied, such as scanning infrastructure with OpenVAS, enabling continuous risk assessment and remediation. While intrusion testing is traditionally manual and periodic, it can complement this continuous model by validating the effectiveness of protections under real-world attack scenarios. In our implementation, SAST tools were used to detect vulnerabilities in the application's source code, while VM targeted infrastructure-level weaknesses.

Our objective was to explore how they might be integrated into a unified and automated security workflow, aiming to promote continuous and comprehensive security throughout the software development life cycle.

Although the current work focuses on the public sector within a governmental institution, the processes and methods discussed are equally applicable to private sector environments, where security demands and development practices often mirror those found in public organizations.

The subsequent subsections detail the techniques, tools, and procedures employed in the current DevSecOps implementation.

3.1 Static Application Security Testing (SAST)

The SAST technique was integrated directly into the CI/CD pipeline, promoting a preventive and continuous security approach in the software development cycle. This integration allows for early detection of flaws, optimizing the cost and time of correction.

The following tools were used for static analysis:

- **Trivy**[5]: A scanner for vulnerabilities in containers, code repositories, and packages.
- **Semgrep**[6]: A customizable static analysis tool applicable to various programming languages.
- **Dependency Check**[7]: Detects publicly disclosed vulnerabilities in project dependencies, utilizing an extensive database (National Vulnerability Database - NVD) for its analysis.

These tools provide detailed and standardized results, with the ability to integrate into CI/CD pipelines, facilitating rapid identification and correction of vulnerabilities throughout development. In addition, they were chosen because they are widely adopted by the development and security communities, including major companies and open-source projects. They are also actively updated to better respond to emerging security threats. Their use in academic and educational settings further reinforces their credibility and technical value.

For the classification of vulnerabilities found by SAST tools, the *Common Vulnerabilities and Exposures* (CVE)[8] and *Common Weakness Enumeration* (CWE)[9] lists were used.

[5] https://github.com/aquasecurity/trivy
[6] https://github.com/semgrep/semgrep
[7] https://github.com/dependency-check/DependencyCheck
[8] https://cve.mitre.org
[9] https://cwe.mitre.org

- **CVE**: Managed by MITRE, this list organizes information on publicly known vulnerabilities, serving as a widely used standard in cybersecurity research and studies [30]. Each vulnerability has a severity score calculated by the CVSS [8].
- **CWE**: Also managed by MITRE, CWE provides information on common types of software weaknesses, along with mitigation strategies and good development practices to avoid introducing these weaknesses [12, 28].

The CVE focuses on documenting specific security flaws in systems and applications that can then each be linked to one or more CWE classifications, which are a more general type of categorization, with one CWE being associated with any CVE that fits the category. This association between the two lists ensures that, when used together, a comprehensive and standardized assessment of flaws is achieved, promoting greater traceability and effective prioritization of fixes.

The phases of a CI/CD pipeline bring together practices and tools aimed at automating the software development process, from code integration to its deployment in production [14, 24]. The secure CI/CD pipeline in this work was designed to integrate the aforementioned SAST tools, ensuring that each code update in the repository triggers an automatic analysis. For the pipeline's execution environment, a local GitLab docker instance was created to run the pipeline on GitLab CI/CD. The integration begins in the Build phase, where static security analysis is performed with the following configurations:

- **Trivy**: Configured to scan the container created in the build phase while filtering for high and critical vulnerabilities. It generates a report in HTML format.
- **Semgrep**: Configured to automatically identify the programming language of code base and use the appropriate ruleset. It generates a report in JSON format.
- **Dependency-Check**: Under default configurations, database download is very slow, so to optimize the download process and reduce latency, an NVD API key was used [17]. It generates a report in HTML format.

Because the pipeline was designed for the three tools to generate their own report, there may be situations where overlapping information appears across the reports. In such cases, the technical team of the government institution will perform a cross-analysis. In addition, all generated reports are set to expire after one day.

3.2 Vulnerability Management (VM)

Vulnerability management followed a cycle of continuous improvement and risk-based prioritization, adapting to the operational constraints and technological environment of the institution. In this work, the main steps were:

1. **Information Collection and Asset Mapping**:

Initially, a collaborative survey was conducted with the institution's technical team to identify available network ranges, the most relevant assets, and appropriate periods for conducting tests. This step is crucial for contextualizing security analyses and efficiently directing efforts. The prioritization of hosts for analysis considered the viability of execution under network constraints, such as firewall rules and access restrictions to servers. Additionally, hosts with associated PTR (Pointer) records were prioritized, as they were more likely to be exposed to external networks and thus represented potentially higher risk. These criteria helped optimize resource allocation while focusing on targets with greater exposure and feasibility of assessment.

2. **Vulnerability Scanning**: The analysis was performed using the open-source tool OpenVAS[10], which is widely recognized in the security community for vulnerability detection. Tests were run in unauthenticated mode, simulating the behavior of an external attacker without valid credentials. This approach allowed us to identify flaws exposed directly on the network surface, respecting operational limitations and avoiding impacts on production systems. These actions were performed outside of business hours to mitigate any impact on the institution's production network.

3. **Vulnerability Analysis, Classification, and Prioritization**: Detected vulnerabilities were classified based on CVSS (Common Vulnerability Scoring System) [9], assigning severity levels (Low, Medium, High and Critical). This classification, automatically provided by OpenVAS based on the Common Vulnerabilities and Exposures (CVE) database, considers aspects such as attack vector, exploitation complexity, required privileges, and impact on confidentiality, integrity, and availability. For more assertive prioritization, the CVSS classification was complemented with the EPSS (Exploit Prediction Scoring System) metric [10], which estimates the actual probability of exploitation for each flaw. EPSS, expressed as a score between 0 and 1, indicates the likelihood that a vulnerability will be exploited by malicious agents within 30 days of its disclosure. Additionally, the existence of a publicly available exploit (software, data block, or exploitation script) in repositories such as Exploit-DB [19] or Metasploit [25] was considered an additional risk factor, as it significantly increases the likelihood that the vulnerability will be exploited by low-sophisticated attackers. The context of the affected asset (function, exposure to Internet, criticality) was also evaluated to refine prioritization.

4. **Consolidation and Continuous Monitoring**: The data generated in the previous steps was consolidated into a document containing an action plan, allowing the agency's technical team to monitor the progress of corrections and implement a continuous remediation cycle.

[10] https://www.openvas.org

4 Results and Discussion

This section presents the results obtained from the case study conducted at SEPLAG, aiming to evaluate the effectiveness of the security practices applied by two distinct but complementary techniques: SAST and VM.

The findings serve as proof of concept, providing an initial assessment of the integration of both approaches into a unified security workflow aligned with DevSecOps principles. This evaluation ensures a solid methodological foundation for subsequent iterations and broader application in real production environments.

4.1 SAST Analysis Results

This section presents the results of applying the CI/CD pipeline with SAST at SEPLAG.

Regarding execution time, the Trivy and Semgrep tools had an execution time of less than 30 seconds each, while the Dependency Check tool, using the NVD API key, took around 3 minutes. Thus, the total execution time of the SAST tools in the CI/CD pipeline was around 4 minutes for a repository with around 23000 lines of code written, making its execution feasible in a real-world scenario and demonstrating that it can be successfully applied to repositories of similar size.

Throughout the analysis, several security vulnerabilities were identified in the application's source code, such as *misconfigurations* and critical flaws, which could be exploited by attackers to gain unauthorized access to sensitive data. A total of 8 vulnerabilities were found during the SAST analysis of an application chosen by SEPLAG, with the following severity distribution (calculated using CVSS): 1 High (CVSS between 7.0 and 8.9), 6 Medium (between 4.0 and 6.9), and 1 Low (between 0.1 and 3.9). This CVSS-based severity categorization was essential for strategically prioritizing fixes, allowing SEPLAG's workforce to be used more effectively. Additionally, Table 1 divides the vulnerabilities found by an assigned number - omitting direct mentions of related CVEs at the request of the institution - their associated CWE and their overall severity, exploitability, and impact scores, calculated using the CVSS v3.1 calculator from NIST [18].

It is noted that most of the vulnerabilities found have an exploitability score lower than 5.0, indicating a low probability of a malicious attack. However, half of the vulnerabilities have an impact score greater than 6.0, with emphasis on vulnerability 01, classified with CWE-200, which has the maximum impact on the application as it is related to the exposure of sensitive information to an unauthorized actor. This fact indicates that, if exploited, this vulnerability could cause significant damage to SEPLAG's assets. These results demonstrate the importance of the CI/CD pipeline with applied SAST, enabling the discovery and correction of vulnerabilities.

The found vulnerabilities can be divided into three categories: (1) Sensitive Data Exposure; (2) Insecure Service Configuration; and (3) Improper Configurations.

Table 1. Vulnerabilities with severity, exploitability, and impact scores.

Vulnerability Number	Common Weakness Enumeration (CWE)	Severity	Exploitability	Impact
01	CWE-200: Exposure of Sensitive Information to an Unauthorized Actor	7.5	5	10
02	CWE-863: Incorrect Authorization	6.8	6.7	6.9
03	CWE-321: Use of Hard-coded Cryptographic Key	6.4	4.8	8
04	CWE-35 Path Traversal	6.3	5.6	7
05	CWE-732: Incorrect Permission Assignment for Critical Resource	4.2	3.4	5
06	CWE-732: Incorrect Permission Assignment for Critical Resource	4.2	3.4	5
07	CWE-209: Generation of Error Message Containing Sensitive Information	4.2	3.4	5
08	CWE-358: Improperly Implemented Security Check for Standard	2	2	2

The first one refers to problems of exposing confidential information in improper locations or to unauthorized users, as in vulnerabilities 01 and 03, classified with CWE-200 and CWE-321 respectively. Furthermore, excessively descriptive error messages, as in vulnerability 07, classified with CWE-209, can reveal sensitive information about the application's structure. To mitigate these risks, confidential data must be stored and managed in secure locations, and for more generic messages to be used to communicate application errors to users.

The second category refers to services used or implemented that were not configured securely, as in the case of vulnerabilities 06 and 07, both classified with CWE-732, where there is incorrect permission assignment, and in the case of vulnerability 08, classified with CWE-358, where there is an improperly implemented security check for a pre-established standard, increasing the risk of application compromise. To mitigate them, it is necessary to remove the incorrect assignments and follow good security practices for the services.

Finally, the third category comprises improper configurations in the application. This includes vulnerability 02, classified with CWE-863, where improper configuration of authorization checks allows unauthorized actions to be performed, as well as vulnerability 04, classified with CWE-35, which indicates the possibility of externally manipulating the path of the file being accessed by the application, allowing access to restricted files and directories. To reduce these risks, it is necessary to reinforce access control and data input validation policies.

Alternatively, we can also divide them using the OWASP Top 10 2021 list because each category is mapped to a set of CWEs [20]. However, not all CWEs are contemplated in OWASP top 10 2021, and thus not all 8 vulnerabilities can be classified using it. In this list, vulnerabilities 01, 02 and 04 are classified as A01:2021 Broken Access Control, while vulnerabilities 03 and 07 are A02:2021 Cryptographic Failures and A04:2021 Insecure Design, respectively. Vulnerabilities 05, 06 and 08's CWE are not mapped to any OWASP Top 10 classification.

From the presented results, although the SAST tools used in this work are already widely known in the industry and thus not innovative by themselves, it is clear that the application of SAST techniques by integrating these tools in SEPLAG's CI/CD pipeline represents an innovation in the context of public administration, which has historically faced challenges implementing security automation and DevSecOps culture. The introduction of these tools represents, when compared to the application's previous pipeline, an improvement in coverage and standardization in vulnerability detection that outweighs the increase in pipeline execution time to around 4 minutes. It reinforces the importance of adopting continuous security practices in the SSDLC, promoting a preventive and sustainable culture of protecting public sector digital assets while adhering to the operational constraints common in the sector. Furthermore, the application of the CI/CD pipeline with SAST directly aligns with international security best practices, such as the NIST CSF and the CIS Controls, by incorporating automated and continuous mechanisms for identifying and mitigating vulnerabilities throughout the development cycle [1].

4.2 VM Analysis Results

The results of the case study presented in this section represent a proof of concept of the VM process, conducted at SEPLAG to obtain results that serve as a basis for validating the approaches and techniques used. This stage aims to ensure the effectiveness of the process before its full application in the real environment analyzed, ensuring greater accuracy and reliability in future results.

The analysis of the identified vulnerabilities is essential to understand the criticality of each one of them, and the CVSS score plays a crucial role in this process. In total, 82 vulnerabilities were identified during the scanning process, with severities (High, Medium, and Low) distributed as follows: 19 High (23.17%), 33 Medium (40.24%) and 30 Low (36.59%). Vulnerabilities classified as high should be treated with top priority, while those of medium and low severity can be scheduled for correction in a medium/long-term mitigation schedule.

OpenVAS identifies several network protocols to detect vulnerabilities in services, including TCP (Transmission Control Protocol), UDP (User Datagram Protocol), and ICMP (Internet Control Message Protocol) for basic communication and host discovery, as well as several application protocols, such as HTTP (Hypertext Transfer Protocol), SSH (Secure Shell), and RDP (Remote Desktop Protocol) [23].

For a more in-depth analysis of asset exposure, Figure 2 shows the distribution of vulnerabilities identified by service port, segmented according to CVSS severity. The five services/ports with the highest number of occurrences were considered, allowing a clear visualization of the criticality associated with each exposed service. This approach allows for the visualization of which services are most vulnerable, facilitating the prioritization of mitigation actions in critical areas.

Among the vulnerabilities with ports identified, critical services stand out, such as SSH (port 22), HTTPS (port 443), and RDP (port 3389), which should

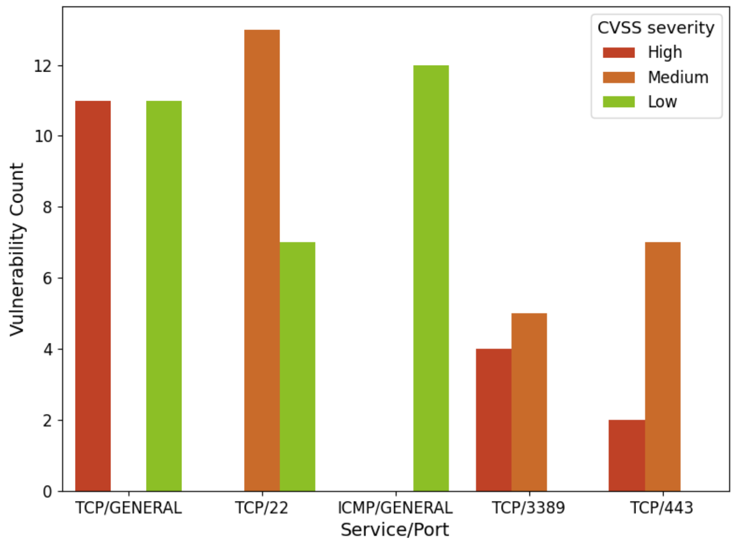

Fig. 2. Analysis by port/service.

be prioritized due to their operational relevance and potential impact in case of exploitation. In addition, vulnerabilities related to Windows network services, such as RPC (Remote Procedure Call) and SMB (Server Message Block) protocols, were also detected, reinforcing the need for specific assessments in these areas. Additionally, OpenVAS may indicate that some occurrences were classified as "GENERAL", which means that the tool was unable to determine the source port of the vulnerability directly. In these cases, it is essential that the reported vulnerability be assessed on the server to identify which ports are effectively impacted. This distribution of vulnerabilities allows mitigation actions to be directed to the most critical services and ports susceptible to attacks, optimizing the use of resources and increasing the effectiveness of the security measures implemented.

Figure 3 presents a joint analysis of the CVSS and EPSS metrics of a set of identified vulnerabilities, allowing a more accurate assessment of both the potential impact and the practical exploitability of each vulnerability. For example, CVEs 3, 4, and 5 are highly critical because they have a High severity (CVSS 9.8, 9.3, and 8.8, respectively) and high exploitability (EPSS 0.975, 0.795, and 0.968, respectively). On the other hand, the vulnerability represented as CVE-1 presented a CVSS of 7.5 (High severity) and an extremely low EPSS (0.001), indicating a technically severe flaw, but with a low probability of exploitation in the current scenario. CVEs 2 and 6 deserve attention because, despite having low severity, they have high exploitability and may be initial or intermediate stages in the chain of events of a cyberattack.

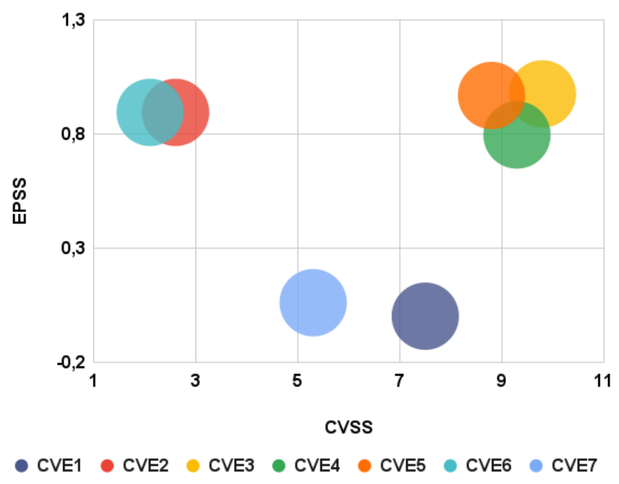

Fig. 3. Analysis by severity and exploitability.

Figure 4 shows the distribution of the number of vulnerabilities associated with each asset identified in the environment. This graph provides a clear view of which assets have the highest number of vulnerabilities. In this analysis, it is important to consider the type and context of the asset, for example, whether it is a server or a common workstation, whether it is exposed to the Internet or not, whether it contains databases that are critical to the organization, etc. This contextualized analysis allows for a correct classification of the risk of vulnerabilities and a more assertive prioritization of the corrections that will need to be made. By analyzing this information, security teams can direct efforts to the most exposed assets, optimizing the use of resources and reducing the risk of critical infrastructure compromises.

The results obtained demonstrate the effectiveness of VM, integrating severity (CVSS), exploitability (EPSS), the existence of public exploits, and the environment context. In addition, it is worth noting that the study aligns with international security best practices, such as the NIST CSF and CIS Controls [1], by structuring a process for identifying and prioritizing vulnerabilities based on risk metrics and operational context. Regarding the NIST CSF, the functions "Identify", "Protect", and "Detect" are used through asset mapping, vulnerability scanning with OpenVAS, and the use of indicators such as CVSS and EPSS. Similarly, about CIS Controls, the highlights are asset inventory controls, continuous vulnerability management, and verification of the existence of public exploits. Thus, the work establishes a solid foundation for the prevention of cyberattacks and more effective incident response plans, aligned with widely recognized frameworks [15].

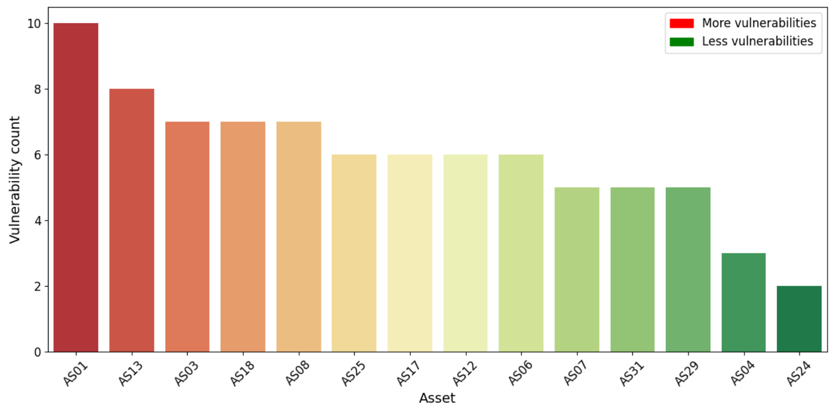

Fig. 4. Total vulnerabilities per asset.

5 Main Findings

This study evaluated the application of structured security techniques within SEPLAG's legacy environment, focusing on two complementary techniques: SAST at the application development level and VM at the infrastructure level. The findings validate the feasibility and effectiveness of these techniques and offer insights into their practical integration into a DevSecOps workflow. We present below the main findings in this research:

Static Application Security Testing (SAST)

- The CI/CD pipeline with integrated SAST tools (Trivy, Semgrep, and Dependency Check) achieved full execution in under 4 minutes, proving viable for real-world DevSecOps adoption.
- A total of 8 code-level vulnerabilities were identified, including 1 High, 6 Moderate, and 1 Low severity.
- Vulnerabilities fell into three main CWE categories: (1) Sensitive Data Exposure; (2) Insecure Service Configuration; and (3) Improper Configurations.
- Despite relatively low exploitability in most cases, several vulnerabilities had high impact scores, posing serious risk if exploited - especially the exposure of sensitive information.
- The approach reinforced international best practices by integrating automated security checks into the software development life cycle (SDLC).

Vulnerability Management (VM)

- A total of 82 vulnerabilities were identified through network scanning using OpenVAS, with the following severity distribution based on CVSS scores: 19 High, 33 Medium, and 30 Low.

- Critical services such as SSH (port 22), HTTPS (port 443), and RDP (port 3389) were among the most exposed, highlighting the need for prioritized mitigation.
- The analysis also incorporated EPSS metrics to assess exploitability, identifying vulnerabilities with high impact and high likelihood of exploitation (e.g., CVEs with CVSS > 8.8 and EPSS > 0.9).
- Asset-level mapping showed an uneven distribution of vulnerabilities, allowing targeted remediation strategies based on operational risk and system criticality.
- The process was aligned with best practices from the NIST CSF and CIS Controls, supporting structured risk-based prioritization.

6 Conclusion

This study demonstrated the feasibility and relevance of applying structured security methodologies to legacy environments in the public sector, where technological diversity and operational constraints demand adaptable and effective cybersecurity strategies. Adopting the DevSecOps methodology becomes crucial for protecting citizen data, keeping services running, and maintaining trust in online public offerings. In this context, two complementary approaches were applied independently in this work: SAST at the development pipeline level and VM at the infrastructure level. The results confirm that addressing vulnerabilities at both levels is essential for improving cyber resilience.

As part of the results, detailed reports were prepared and delivered to the management team, outlining the main findings from each individual analysis. These reports included specific mitigation recommendations for the vulnerabilities identified, as well as a proposed action plan to ensure continuity of security assessments. This plan is intended to support strategic decision-making, enabling managers to prioritize high-risk issues and take immediate corrective measures when necessary.

Although DAST was not implemented in this work due to limitations in SEPLAG's current infrastructure, its importance in runtime security analysis is well recognized. Future work intends to expand the security pipeline to include DAST, further aligning with DevSecOps principles and strengthening SEPLAG's proactive cybersecurity posture.

Acknowledgment

The authors would like to thank CNPq (Processes 306362/2021-0 and 303877/2021-9) and FUNCAP/SEPLAG, through the Chief Scientist program in the project "Vulnerability Testing and Cybersecurity Monitoring in SEPLAG Computer Systems and Networks", for the financial support.

References

1. Bashofi, I., Salman, M.: Cybersecurity maturity assessment design using NIST CSF, CIS Controls v8 and ISO/IEC 27002. In: 2022 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom). pp. 58–62. IEEE (2022)
2. Bower, L.: Brazil as a leader in digital transformation. In: Proceedings of the 16th international conference on theory and practice of electronic governance. pp. 80–85 (2023)
3. Center for Internet Security: CIS Critical Security Controls v8. https://www.cisecurity.org/controls/cis-controls-list (2023), accessed: 2025-07-03
4. Darus, M.Y., Bolhan, M.F.B., Kurniawan, A., Muliono, Y., Pardomuan, C.R., Hata, M.M.: Enhancing web application penetration testing with a static application security testing (SAST) tool. In: 2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE). pp. 1–6. IEEE (2023)
5. Díaz, O., Muñoz, M., Mejía, J.: Responsive infrastructure with cybersecurity for automated high availability devsecops processes. In: 2019 8th International Conference On Software Process Improvement (CIMPS). pp. 1–9 (2019). https://doi.org/10.1109/CIMPS49236.2019.9082439
6. Efendi, M., Raharjo, T., Suhanto, A.: Devsecops approach in software development case study: Public company logistic agency. In: 2021 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS. pp. 96–101 (2021). https://doi.org/10.1109/ICIMCIS53775.2021.9699316
7. Ficco, M., Granata, D., Palmieri, F., Rak, M.: A systematic approach for threat and vulnerability analysis of unmanned aerial vehicles. Internet of Things **26**, 101180 (2024)
8. FIRST: Forum of Incident Response and Security Teams - Common Vulnerability Scoring System v3.1: Specification Document (2019), https://www.first.org/cvss/specification-document, acessado em: 17 maio 2025
9. FIRST: Forum of Incident Response and Security Teams - Common Vulnerability Scoring System v3.1: Specification Document (2019), https://www.first.org/cvss/specification-document, acesso em: 17 maio 2025
10. FIRST: Forum of Incident Response and Security Teams - Exploit Prediction Scoring System (EPSS) – v2 Model Documentation (2022), https://www.first.org/epss/model, acesso em: 17 maio 2025
11. Ibrahim, A., Valli, C., McAteer, I., Chaudhry, J.: A security review of local government using nist csf: a case study. The Journal of Supercomputing **74**, 5171–5186 (2018)
12. Kota, K., Manjunatha, A., et al.: CWE prediction using CVE description-the semantic similarity approach. Procedia Computer Science **235**, 1167–1178 (2024)
13. Lanza, B.B.B., Ávila, T.J.T., Valotto, D.: An overview of rede.gov.br as a federative mechanism for digital government development in Brazil. In: Proceedings of the 23rd Annual International Conference on Digital Government Research. pp. 380–390 (2022)
14. Marandi, M., Bertia, A., Silas, S.: Implementing and automating security scanning to a devsecops ci/cd pipeline. In: 2023 World Conference on Communication and Computing (WCONF). pp. 1–6 (2023). https://doi.org/10.1109/WCONF58270.2023.10235015
15. Möller, D.P.: NIST cybersecurity framework and MITRE cybersecurity criteria. In: Guide to Cybersecurity in Digital Transformation: Trends, Methods, Technologies, Applications and Best Practices, pp. 231–271. Springer (2023)

16. National Institute of Standards and Technology (NIST): Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1. Tech. rep., NIST (April 2018), https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf, acesso em: 17 maio 2025
17. NIST: NVD API: keys, documentation, and request limits. National Institute of Standards and Technology. https://nvd.nist.gov/general/news/API-Key-Announcement (2023), acessado em: 25 de Maio de 2025.
18. NIST: Common vulnerability scoring system (CVSS) calculator. National Institute of Standards and Technology. https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator (2025), acessado em: 17 de maio de 2025
19. Offensive Security: Exploit Database (Exploit-DB) (2025), https://www.exploit-db.com, acesso em: 17 maio 2025
20. OWASP: Owasp top 10. Open Web Application Security Project https://owasp.org/Top10/ (2021), accessed: 2025-07-04.
21. Pimenta, I., Silva, D., Moura, E., Silveira, M., Gomes, R.L.: Impact of data anonymization in machine learning models. In: Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing. p. 188–191. LADC '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3697090.3699865, https://doi.org/10.1145/3697090.3699865
22. Prabowo, S., Putrada, A.G., Oktaviani, I.D., Abdurohman, M., Janssen, M., Nuha, H.H., Sutikno, S.: Privacy-preserving tools and technologies: Government adoption and challenges. IEEE Access (2025)
23. Rahalkar, S.: OpenVAS. In: Quick Start Guide to Penetration Testing: With NMAP, OpenVAS and Metasploit, pp. 47–71. Springer (2018)
24. Rangnau, T., Buijtenen, R.v., Fransen, F., Turkmen, F.: Continuous security testing: A case study on integrating dynamic security testing tools in CI/CD pipelines. In: 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC). pp. 145–154. IEEE (2020)
25. Rapid7: Metasploit Framework (2025), https://www.metasploit.com, acesso em: 17 maio 2025
26. Riaz, S., Asif, A., Khan, Y., Ibrar, M., Afzal, S., Hamid, K., Gul, S., Iqbal, M.W.: Software development empowered and secured by integrating a devsecops design. Journal of Computing and Biomedical Informatics **8**(02) (2025)
27. Safitra, M.F., Lubis, M., Widjajarto, A.: Security vulnerability analysis using penetration testing execution standard (PTES): case study of government's website. In: Proceedings of the 2023 6th international conference on electronics, communications and control engineering. pp. 139–145 (2023)
28. Santos, J.C., Tarrit, K., Sejfia, A., Mirakhorli, M., Galster, M.: An empirical study of tactical vulnerabilities. Journal of Systems and Software **149**, 263–284 (2019)
29. Serasa Experian: Relatório de identidade digital e fraude (2025), edição Especial Segmento Financeiro
30. Wang, T., Qin, S., Chow, K.P.: Towards vulnerability types classification using pure self-attention: A common weakness enumeration based approach. In: 2021 IEEE 24th international conference on computational science and engineering (CSE). pp. 146–153. IEEE (2021)
31. Zheng, Y., Li, Z., Xu, X., Zhao, Q.: Dynamic defenses in cyber security: Techniques, methods and challenges. Digital Communications and Networks **8**(4), 422–435 (2022)