RESEARCH ARTICLE

Multicloud Security Assessment: A Benchmark Study of Infrastructure as Code Scanners

Harry Roe[1], **Mandar Gogate**[1] **and Kia Dashtipour**[1,*]

[1] School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, United Kingdom

Abstract

Multicloud environments are becoming more common, often businesses will have workloads across one or more of AWS, Azure and GCP, with each provider slightly differing in security features and capabilities. Furthermore, Infrastructure as Code is increasing in popularity meaning cloud resources are being provisioned as code through automation pipelines as opposed to GUI/Portal deployments. This shift means that security scanning of the resource code is a crucial first step in securing a cloud environment, and the tool(s) being used for this need to be able to perform at a consistent level across all the different cloud providers. Failure to do this could mean the introduction of security vulnerabilities to the environment, possibly vulnerabilities that have been caught for one cloud provider but not another. This study analyses three well-used Infrastructure as Code scanners when used in resource deployments to the three major cloud providers: AWS, Azure and GCP. The experiment was performed in an isolated CI pipeline to mirror a production workload and used intentionally vulnerable code to give the tools a benchmark number of findings. The findings show a difference in performance for the three tools based on the cloud provider, proving emphatically the importance of understanding all default security controls in a cloud environment and how they can differ based on the provider, as well as the rule coverage for the tools being considered. The findings of this project can be used to give professionals a more informed opinion when choosing one or more of these security scanners.

Keywords: multicloud, infrastructure, code scanners.

1 Introduction

Cloud-hosted applications continue to dominate modern software markets, resulting in an increasing number of organisations migrating infrastructure and workloads to the cloud. As adoption grows, cybersecurity teams are required to understand and advise on secure deployment practices across a wide range of resource types and multiple cloud service providers[1, 2].

A significant focus within contemporary cloud security is Infrastructure as Code (IaC), particularly the risk that insecure deployments may introduce vulnerabilities or misconfigurations that impact the confidentiality, integrity, or availability of cloud systems and data. IaC enables infrastructure to be provisioned programmatically through

Submitted: 13 February 2026
Accepted: 31 March 2026
Published: 29 May 2026

Vol. 2, **No.** 2, 2026.
10.62762/TISC.2026.777114

***Corresponding author:**
✉ Kia Dashtipour
k.dashtipour@napier.ac.uk

Citation
Roe, H., Gogate, M., & Dashtipour, K. (2026). Multicloud Security Assessment: A Benchmark Study of Infrastructure as Code Scanners. *ICCK Transactions on Information Security and Cryptography*, 2(2), 109–118.

automation pipelines rather than manual portal-based configuration, increasing both deployment speed and scale. However, this shift also amplifies the potential impact of configuration errors, as insecure patterns can be rapidly propagated across environments.

Securing IaC deployments is commonly achieved through static security scanning tools and the integration of continuous security controls within CI/CD pipelines. These approaches align with DevSecOps principles, promoting the early identification of security issues during development rather than post-deployment. Despite this, questions remain regarding the effectiveness and consistency of these tools, particularly in multi-cloud environments where cloud service providers differ in both default security configurations and supported capabilities [3, 4].

Multi-cloud adoption presents additional complexity for security practitioners. Differences between what providers consider a baseline security configuration, and which controls are enabled by default, can influence both the presence of vulnerabilities and the ability of security scanners to detect them. As a result, a security issue may be identified in one cloud environment but remain undetected in another, despite identical IaC templates being used. This inconsistency introduces risk for organisations operating across multiple providers.

This research evaluates the performance of three widely used IaC security scanning tools—Checkov, tfsec, and Terrascan—when applied to intentionally vulnerable IaC deployments targeting Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). The tools are assessed within a controlled CI pipeline designed to mirror real-world DevSecOps workflows, using identical misconfigurations across all three environments to provide a consistent basis for comparison [5].

The primary objective of this study is to determine how effectively these tools detect security misconfigurations in a multi-cloud context, and whether detection accuracy varies based on the chosen cloud service provider or the severity of the underlying issue. By analysing detection coverage and consistency across providers, this research aims to provide practitioners with clearer insight into the strengths and limitations of commonly adopted IaC security scanners, supporting more informed tool selection and secure-by-design cloud deployments [6]. The goal is to answer the following research questions:

- Which security scanner provides the best protection and best performance regardless of the cloud environment?
- Does the effectiveness/accuracy of findings with each security scanner differ based on the chosen cloud service provider?
- Will the severity of the misconfiguration have an impact on the detection accuracy?

2 Related Work

This literature review examines the evolution of cloud security risks with a focus on Infrastructure as Code (IaC), the challenges associated with securing IaC deployments, and the role of DevSecOps and security scanning tools in mitigating these risks. Existing research is critically analysed to identify limitations in current approaches and to highlight gaps relevant to multi-cloud environments.

2.1 Evolution of Cloud Security Risks

Cloud computing provides organisations with fast, scalable, and geographically distributed resources that can be provisioned and deallocated on demand. This shift has transformed infrastructure deployment models, with many organisations replacing on-premises hardware with cloud-hosted services. However, this transition has introduced new security risks that require careful consideration.

Recent industry research identifies misconfiguration and inadequate change control as the most significant cloud security threats, surpassing traditional concerns such as identity compromise. While influential, these findings rely on self-reported data and may therefore reflect only those issues that organisations are aware of, rather than those that remain undetected. This highlights the need for proactive security mechanisms capable of identifying misconfigurations before deployment [7].

2.2 Infrastructure as Code and Security Challenges

Infrastructure as Code is widely adopted due to its ability to automate deployments, enforce consistency, and enable version control. By defining infrastructure in code, organisations can rapidly deploy identical resources across environments, significantly improving operational efficiency. However, IaC does not inherently enforce secure configurations, allowing vulnerabilities to be embedded directly into deployment templates.

Empirical studies analysing large volumes of publicly available Terraform templates have identified high rates of security misconfiguration, with nearly half containing issues that could lead to exploitable vulnerabilities. While these studies demonstrate the prevalence of insecure IaC practices, their reliance on static repositories introduces limitations. Public repositories may contain incomplete, outdated, or non-production code, making it difficult to assess the real-world impact of identified issues.

Interest in IaC has grown steadily since 2015, with a notable increase between 2020 and 2022, coinciding with the mainstream adoption of Terraform [3, 8], as illustrated in Figure 1.

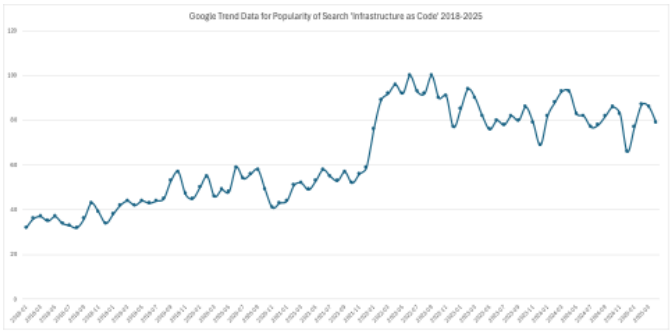

Figure 1. 'Infrastructure as Code' Google Search Popularity Over Time..

In contrast, interest in IaC security has remained comparatively low, despite a sharp rise beginning in 2020, as shown in Figure 2.

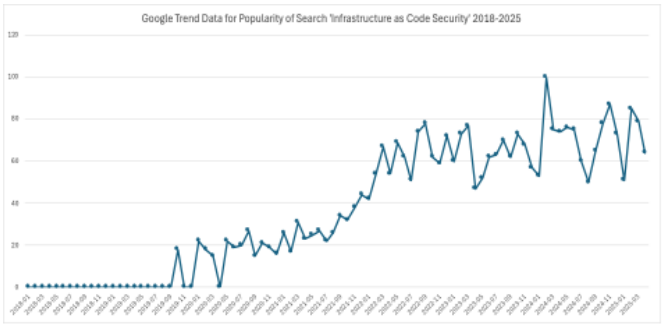

Figure 2. 'Infrastructure as Code Security' Google Search Popularity Over Time.

When these trends are compared (Figure 3), security-focused interest represents only a small proportion of overall IaC interest, reinforcing concerns raised in earlier research regarding the lack of security-focused frameworks and guidance.

Although Google Trends data does not directly measure industry adoption or practitioner awareness, the alignment between increased interest and

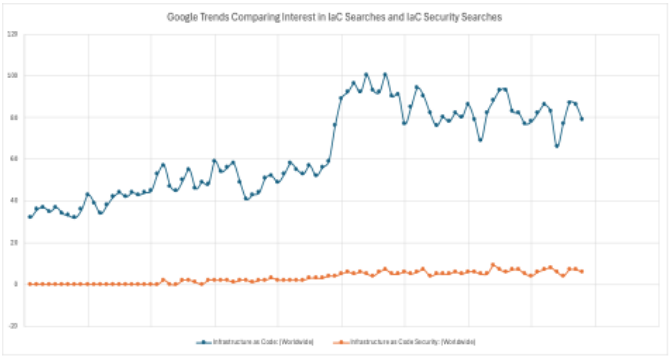

Figure 3. Search Popularity Comparison.

Terraform's growth strengthens the relevance of these observations. Collectively, these findings suggest that IaC security remains under-prioritised relative to the rapid expansion of IaC adoption [9].

2.3 DevSecOps and Shift-Left Security

The growing security risks associated with IaC raise questions regarding responsibility for securing deployments. DevSecOps has emerged as the industry response, promoting the integration of security practices throughout the development lifecycle rather than treating security as a final review stage [10].

DevSecOps is not a specific team or tool but a cultural and procedural shift that extends DevOps principles to include security. It emphasises continuous integration, automation, and early security testing, enabling vulnerabilities to be identified before deployment [3], as conceptualised in Figure 4.

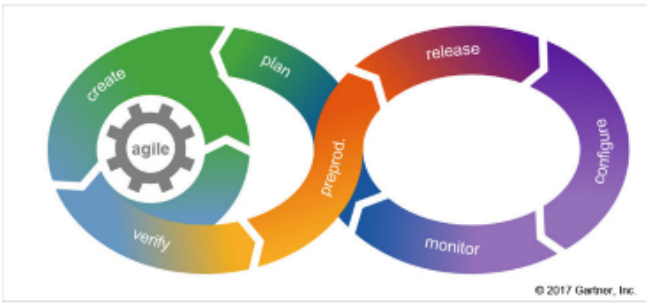

Figure 4. Gartner DevOps Model.

Shifting security controls to earlier stages of development introduces operational challenges, particularly in organisations with clearly defined development, operations, and security silos. Studies highlight that poor communication between these teams can lead to misconfigurations and security incidents, while cultural resistance and skills gaps can hinder DevSecOps adoption.

Despite these challenges, research consistently

indicates that integrating security early improves security posture, efficiency, and compliance. Automation and tooling are frequently recommended as mechanisms to reduce reliance on specialist security knowledge, enabling developers to identify insecure patterns without extensive retraining [11, 12]. A typical DevSecOps toolchain integrating these controls is illustrated in Figure 5.

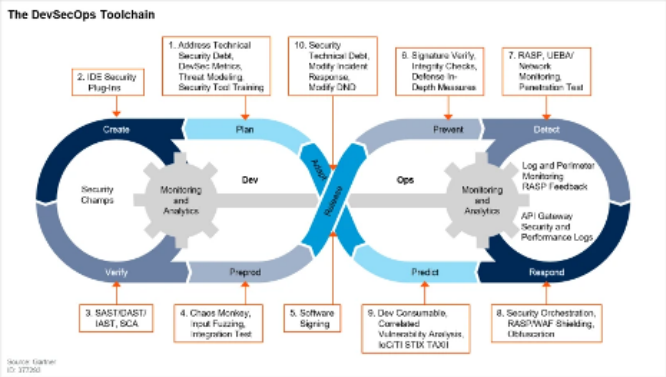

Figure 5. DevSecOps Toolchain.

2.4 IaC Security Posture and Tooling

Effective DevSecOps adoption requires careful selection of cloud service providers, IaC deployment tools, and security scanners. The three dominant cloud providers—AWS, Azure, and GCP—account for the majority of global cloud infrastructure usage (Figure 6), each offering differing security defaults and configuration models.

While each provider offers native IaC solutions, their differing syntax, feature support, and long-term viability pose challenges for multi-cloud environments, as summarised in Table 1. Terraform's cloud-agnostic design and broad provider support have led to widespread adoption, particularly in organisations operating across multiple cloud platforms.

Static Application Security Testing (SAST) tools are commonly used to analyse IaC templates prior to deployment, identifying insecure patterns, misconfigurations, and policy violations without executing the code. Studies evaluating IaC security

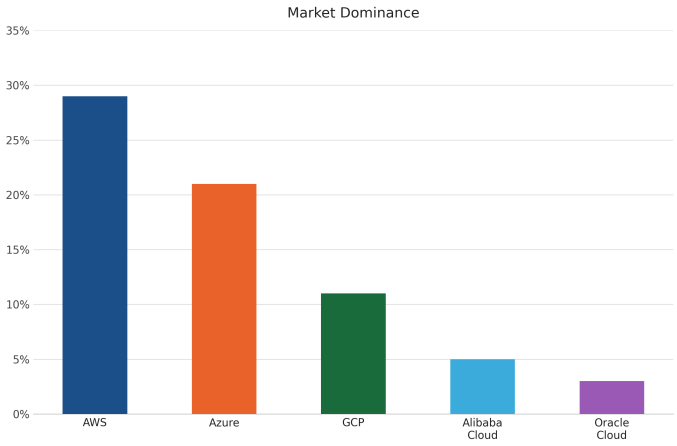

Figure 6. Global Market Share of Major Cloud Providers.

scanners report varying effectiveness depending on the tool, cloud provider, and configuration complexity. Checkov is frequently cited for its extensive policy coverage, while tfsec is noted for its performance and ease of use. Terrascan offers broader IaC language support but has been observed to have a steeper learning curve and less consistent detection coverage [5, 13–15].

A recurring limitation across these tools is their reliance on static analysis. They lack awareness of provider-specific defaults, runtime behaviour, and contextual factors that may render certain configurations redundant or non-exploitable. As a result, false positives and missed detections remain common, necessitating manual review [16].

Existing research demonstrates the value of IaC security scanning tools while also highlighting limitations related to false positives, provider bias, and inconsistent coverage. However, most studies evaluate a limited subset of tools, focus on a single cloud provider, or rely on heterogeneous datasets derived from public repositories.

There is a clear absence of controlled evaluations that assess multiple IaC security scanners across AWS, Azure, and GCP using identical misconfigurations deployed through a CI/CD pipeline. This gap limits the ability of practitioners to understand how

Table 1. Comparison of IaC Tools.

Tool	Provider	Language	Uses
CloudFormation	AWS	YAML, JSON	Deploying AWS resources
Bicep	Azure	DSL	Deploying Azure resources
Deployment Manager	GCP	YAML, Python	Deploying GCP resources
Terraform	HashiCorp	HashiCorp Configuration Language (HCL)	Deploying resources to single or multicloud environments

tool performance varies in real-world multi-cloud environments.

This study addresses this gap by conducting a controlled, repeatable evaluation of Checkov, tfsec, and Terrascan across all three major cloud providers, using uniform misconfigurations and a simulated production CI pipeline.

3 Proposed Methodology

This study evaluates the effectiveness of selected Infrastructure as Code (IaC) security scanning tools across Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). The tools are assessed within a controlled GitLab CI pipeline using intentionally vulnerable Terraform templates, designed to closely mirror real-world DevSecOps deployment workflows.

3.1 IaC Templates and Misconfigurations

Terraform was selected as the IaC tool due to its cross-cloud compatibility and support across all three target cloud providers. Custom Terraform templates were developed for each provider, with efforts made to ensure functional and structural consistency across deployments.

Intentionally vulnerable configurations were introduced into each template based on eight misconfiguration categories derived from prior work and informed by industry benchmarks, including CIS Foundations Benchmarks and cloud provider security best practices. These misconfigurations were selected to represent realistic security risks and were ordered according to severity and likelihood of occurrence in real-world deployments, as detailed in Table 2.

Virtual machines were chosen as the target resource type due to their configurational flexibility and relevance across AWS, Azure, and GCP. While the templates were designed to be as uniform as possible, some provider-specific differences were unavoidable due to variations in default security controls, such as encryption behaviour and networking constraints.

3.2 Security Scanning Tools

Three widely used static IaC security scanners were selected for evaluation: Checkov, tfsec, and Terrascan. These tools were chosen based on their prevalence in existing research and industry adoption, as well as their support for Terraform. All tools were installed using their recommended methods and configured to use the latest stable versions available at the time of testing. To ensure consistency across pipeline executions, the tools were packaged into a custom Docker image based on a minimal Debian Python image. This approach ensured uniform execution environments across all CI runs.

3.3 CI Pipeline Design

All testing was conducted within a GitLab CI environment to simulate a production-style DevSecOps workflow. Cloud credentials for AWS, Azure, and GCP were securely stored as CI/CD variables and injected at runtime to avoid hardcoding sensitive information. The overall multicloud deployment architecture is illustrated in Figure 7.

The pipeline consisted of six stages: initialisation, validation, planning, scanning, deployment, and teardown, as shown in Figure 8. While the deployment and destruction stages were not strictly required for static analysis, they were included to reflect realistic

Table 2. Template Misconfigurations for Testing.

Risk Rank	Misconfiguration	Associated Risk
Critical	Admin by default	Lateral movement with uncontrolled access to services
Critical	IP address binding	Increase external attack surface & risk of brute forcing
High	IAM misconfig	Unauthorised access leading to privilege escalation
High	Hardcoded secrets	Credential leakage
Medium	No encryption in transit	Data interception, open to man-in-the-middle attacks
Medium	No encryption at rest	Data exposure
Low	Outdated feature	Exploitable vulnerabilities
Low	No logging or monitoring	Lack of forensic trail

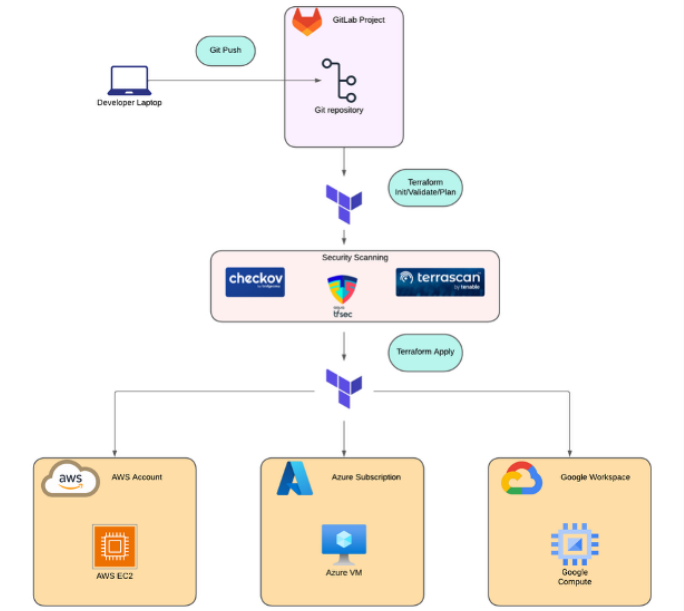

Figure 7. Multicloud IaC Deployment for Experiment.

CI/CD practices and ensure end-to-end workflow integrity.

During the scanning stage, each IaC template was evaluated independently by all three security tools. Scan outputs were exported as pipeline artefacts for further analysis.

3.4 Evaluation Metrics

Evaluation focused on three key criteria aligned with the research questions: detection coverage, cross-provider consistency, and severity alignment. Scanner outputs were manually reviewed to identify true positives, false positives, and false negatives, with detections mapped directly to the intentionally introduced misconfigurations. To ensure result reliability, the pipeline was executed multiple times, with identical results observed across runs. All findings were normalised into a human-readable format to support comparative analysis between tools and cloud providers.

This study did not involve human participants, production data, or live enterprise infrastructure. All cloud resources were deployed within isolated test environments and destroyed immediately following analysis to prevent unintended exposure or financial impact. Access to the CI environment, IaC templates, and cloud resources was restricted to authorised project contributors. Credentials were managed using least-privilege principles and stored securely within the CI system. These controls ensured that the intentionally insecure infrastructure posed no risk beyond the scope of the experiment.

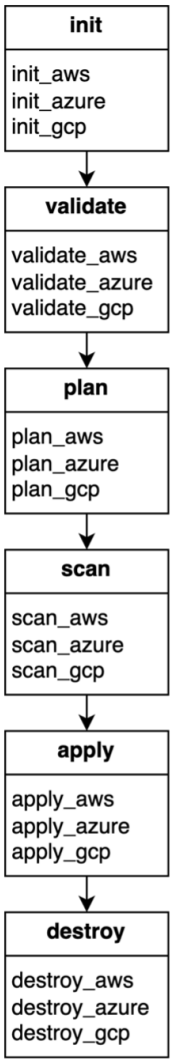

Figure 8. Stages of GitLab CI Pipeline.

4 Results

The testing phase was executed successfully across all three cloud providers, producing a consistent and repeatable set of results. All pipeline executions yielded identical findings, confirming the reliability of the experimental setup. The results presented in this section address each research question through quantitative analysis of scanner detections across AWS, Azure, and GCP. Full, unfiltered scan outputs for each tool and cloud provider are provided in the appendix. This section focuses on aggregated results and comparative patterns relevant to the research objectives.

4.1 Overall Scanner Performance (RQ1)

Across all deployments, Checkov consistently reported the highest number of findings, followed by tfsec and then Terrascan. This pattern was observed for AWS, Azure, and GCP, although the total number of detections varied by provider, as presented in Table 3.

Table 3. Number of findings for each tool per cloud provider.

	AWS	Azure	GCP	Total
Checkov Findings	9	7	8	24
tfsec Findings	6	3	6	15
Terrascan Findings	3	2	1	6

Checkov detected between seven and nine misconfigurations per deployment, while tfsec reported between three and six. Terrascan produced the lowest number of detections overall, identifying three or fewer issues in any individual deployment. Despite identical misconfigurations being present in all templates, no tool reported a consistent number of findings across all three cloud providers. This indicates that none of the evaluated scanners provided uniform detection coverage in a multicloud context.

4.2 Impact of Cloud Provider on Detection Accuracy (RQ2)

Detection accuracy varied noticeably depending on the target cloud service provider. When aggregating findings across all tools, AWS deployments produced the highest total number of detections, followed by GCP, with Azure yielding the lowest, as shown in Figure 9.

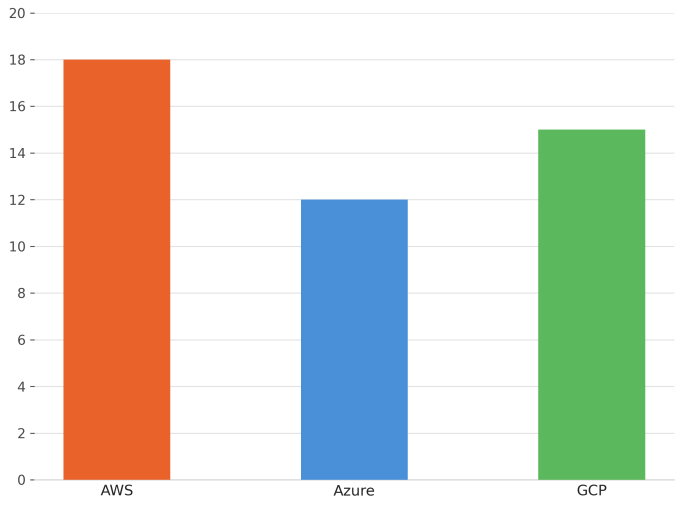

Figure 9. Total detections per cloud provider.

All three scanners demonstrated stronger detection performance when analysing AWS templates. In contrast, Azure deployments consistently resulted in fewer reported issues, even though identical vulnerabilities were present. GCP results fell between AWS and Azure but showed particularly low detection counts when scanned with Terrascan. These findings indicate that scanner effectiveness is influenced by the underlying cloud provider, likely due to differences in default security controls and how configuration options are exposed through Terraform.

4.3 Detection by Severity (RQ3)

To evaluate whether misconfiguration severity influenced detection accuracy, findings were grouped according to the predefined risk rankings assigned during template design. Detections were categorised as Critical, High, Medium, Low, or Other, where "Other" represents findings not directly related to the intentionally introduced misconfigurations, as illustrated in Figure 10.

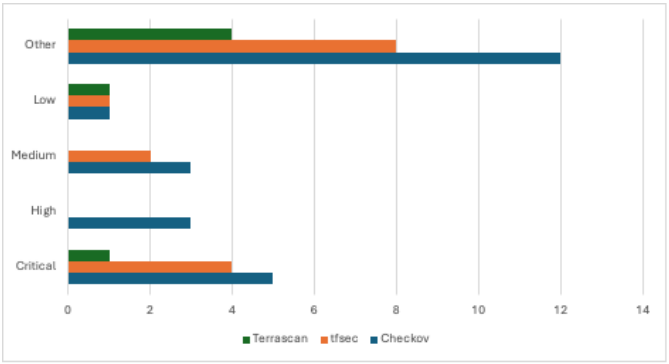

Figure 10. Findings based on severity.

Critical misconfigurations, particularly those related to identity and access management, were the most consistently detected across tools. Checkov was the only scanner to report findings across all severity categories. tfsec detected several Critical and Medium issues but did not identify any High severity misconfigurations. Terrascan detected only a small number of Critical and Low severity issues and failed to identify any Medium or High severity misconfigurations. A significant proportion of reported findings across all tools fell into the “Other” category. These findings were generally lower severity and not directly associated with the benchmarked misconfigurations, indicating additional noise within the scan outputs.

Overall, detection accuracy decreased as misconfiguration severity moved away from identity-related risks, with several intentionally introduced issues remaining undetected regardless of

severity.

5 Discussion

The results of this study demonstrate that static Infrastructure as Code (IaC) security scanners exhibit notable variation in detection coverage across both tools and cloud service providers. Although all three scanners were applied to identical Terraform templates containing the same intentional misconfigurations, none produced consistent results across AWS, Azure, and GCP. This finding reinforces concerns raised in prior research regarding provider bias and inconsistent rule coverage in static IaC security tools.

Across all deployments, Checkov consistently produced the highest number of findings, particularly in AWS environments. This aligns with previous studies that identify Checkov as offering the most comprehensive policy coverage among commonly used IaC scanners. However, the increased number of detections was accompanied by a higher volume of findings unrelated to the intentionally introduced misconfigurations. While broader coverage may be beneficial for identifying a wider range of potential issues, it also increases the likelihood of alert fatigue, particularly in environments where security teams must triage large volumes of scan output.

tfsec demonstrated a more selective detection profile, producing fewer findings overall and generating less noise. This behaviour mirrors observations from earlier evaluations, where tfsec was noted for its efficiency and ease of use but also for its comparatively narrower policy scope. In this study, tfsec failed to detect all high-severity misconfigurations, suggesting that reduced noise may come at the cost of reduced coverage. Terrascan showed the weakest performance, failing to detect the majority of intentionally introduced issues across all providers. Notably, Terrascan did not identify any misconfigurations that were not also detected by either Checkov or tfsec, calling into question its practical value in a multicloud Terraform-based workflow.

The influence of cloud provider on detection accuracy was pronounced. AWS deployments consistently produced the highest number of findings across all tools, while Azure yielded the fewest. This disparity is best explained by differences in provider defaults and configuration exposure rather than inherent security differences between platforms. Azure enforces stricter default controls, such as mandatory encryption at rest, which reduce the number of detectable misconfigurations available to static scanners. GCP imposes intermediate-level defaults, resulting in detection counts between those of AWS and Azure. In contrast, AWS exposes a greater number of configurable parameters, increasing the likelihood that scanner rules will trigger. Similar trends have been reported in previous multicloud evaluations, strengthening the argument that scanner effectiveness is tightly coupled to provider design rather than purely tool capability [13].

Severity analysis further highlights limitations in static IaC scanning. Critical misconfigurations related to identity and access management were the most consistently detected across all tools, reflecting strong policy coverage in this area. However, several high and medium severity issues, including hardcoded secrets, encryption in transit, and outdated software usage, were frequently missed. The failure of all three tools to detect certain intentionally introduced misconfigurations indicates structural blind spots in static analysis approaches, particularly where vulnerabilities depend on contextual or runtime behaviour.

These findings support the growing consensus that static IaC scanners, while valuable, are insufficient as a standalone security control. Their reliance on predefined rule sets limits their ability to adapt to evolving cloud services and nuanced configuration contexts. As a result, organisations adopting DevSecOps practices should treat IaC scanning as one layer within a broader security strategy, supplemented by manual review, custom policy development, and post-deployment monitoring.

While this study was designed to maximise experimental control, several limitations should be acknowledged. The evaluation focused on a single resource type and a limited set of misconfigurations, which may not fully represent the complexity of enterprise-scale IaC deployments. Additionally, only static analysis was considered; dynamic or runtime security tools may identify issues that static scanners cannot. Despite these limitations, the controlled and repeatable nature of the experiment provides clear insight into how tool and provider choice influences IaC security outcomes in multicloud environments.

6 Conclusion and Future Work

This study evaluated the effectiveness of three widely adopted Infrastructure as Code (IaC) security scanning tools—Checkov, tfsec, and Terrascan—across

Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). Using intentionally vulnerable Terraform templates deployed through a controlled continuous integration pipeline, the research assessed whether detection accuracy and coverage vary based on tool selection, cloud provider, and misconfiguration severity.

The findings demonstrate that static IaC scanner performance is strongly influenced by both the chosen tool and the target cloud provider. Checkov consistently identified the highest number of misconfigurations across all environments, particularly in AWS deployments, reflecting its broader policy coverage. tfsec produced fewer findings with reduced noise but failed to detect several higher-severity issues, while Terrascan exhibited limited detection capability across all providers. No single tool provided complete or consistent coverage across all three cloud platforms.

Cloud provider design also played a significant role in detection outcomes. AWS deployments yielded the highest number of findings, whereas Azure consistently produced the lowest, largely due to stricter default security controls. These results highlight that detection volume alone should not be interpreted as an indicator of security posture, but rather as a reflection of how provider defaults interact with scanner rule sets.

Severity-based analysis further revealed structural limitations in static IaC scanning. While critical identity-related misconfigurations were reliably detected, several high and medium severity issues remained undetected across all tools. This reinforces the need to treat static IaC scanning as a complementary control rather than a standalone security solution.

Overall, this research contributes a controlled, multicloud evaluation of IaC security scanners using uniform misconfigurations and a realistic CI pipeline, addressing a gap in existing literature. The findings emphasise the importance of combining multiple scanning tools with provider-specific knowledge, custom policies, and additional security controls to support secure-by-design cloud infrastructure in multicloud environments.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. *Information and software technology, 141*, 106700. [CrossRef]

[2] Olowookere, A., Gogate, M., Hussain, A., Asim, M., Babar, M., Hussain, A., & Dashtipour, K. (2026). Evaluation of decision tree-based ensemble learning models in obfuscated malware detection and classification. In *Cybersecurity, Cybercrimes, and Smart Emerging Technologies* (pp. 65-71). CRC Press.

[3] Rahman, A., Williams, L., Snipes, W., & Slankas, J. (2019). Infrastructure as code: Security challenges and best practices. In *2019 IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 271-280). IEEE. [CrossRef]

[4] Verdet, A., Hamdaqa, M., Silva, L. D., & Khomh, F. (2025). Assessing the adoption of security policies by developers in terraform across different cloud providers. *Empirical Software Engineering, 30*(3), 74. [CrossRef]

[5] Konala, P. R. R., Kumar, V., & Bainbridge, D. (2023, July). SoK: static configuration analysis in infrastructure as code scripts. In *2023 IEEE international conference on cyber security and resilience (CSR)* (pp. 281-288). IEEE. [CrossRef]

[6] Chiari, M., De Pascalis, M., & Pradella, M. (2022, March). Static analysis of infrastructure as code: a survey. In *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)* (pp. 218-225). IEEE. [CrossRef]

[7] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM, 53*(4), 50-58. [CrossRef]

[8] Myrbakken, H., & Colomo-Palacios, R. (2017, September). DevSecOps: a multivocal literature

review. In *International Conference on Software Process Improvement and Capability Determination* (pp. 17-29). Cham: Springer International Publishing. [CrossRef]

[9] Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. *ACM Computing Surveys, 52*(6), 1-35. [CrossRef]

[10] Singer, P. W., & Friedman, A. (2013). *Cybersecurity and Cyberwar: What Everyone Needs to Know®*. Oxford University Press.

[11] Rahman, A., & Williams, L. (2020). Security smells in infrastructure as code. *Empirical Software Engineering, 25*(1), 1-44. [CrossRef]

[12] National Institute of Standards and Technology. (2022). Secure Software Development Framework (SSDF). *NIST Special Publication 800-218*. [CrossRef]

[13] Fernandes, D. A., Soares, L. F., Gomes, J. V., Freire, M. M., & Inácio, P. R. (2014). Security issues in cloud environments: a survey. *International journal of information security, 13*(2), 113-170. [CrossRef]

[14] Rahman, A., Mahdavi-Hezaveh, R., & Williams, L. (2019). A systematic mapping study of infrastructure as code research. *Information and Software Technology, 108*, 65-77. [CrossRef]

[15] Rahman, A., & Williams, L. (2020). Characterizing infrastructure as code security in public repositories. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 362-372). IEEE. [CrossRef]

[16] Ayewah, N., & Pugh, W. (2008). Using static analysis to find bugs. *IEEE Software, 25*(5), 22-29. [CrossRef]