Análise de Vulnerabilidades em Configurações Padrão de Serviços em Provedores de Computação em Nuvem

Caroline de Oliveira Braga[1], **Dalbert Matos Mascarenhas**[1], **Igor M. Moraes**[2]

[1]Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - CEFET/RJ
Petrópolis - RJ - Brasil

[2]Laboratório MidiaCom – IC/TCC/PGC
Universidade Federal Fluminense (UFF), Niterói – RJ – Brasil

caroline.braga@aluno.cefet-rj.br,
dalbert.mascarenhas@cefet-rj.br, igor@ic.uff.br,

Abstract. *This work presents a comparative analysis of security vulnerabilities originating from default settings in major cloud providers (AWS, Azure, and GCP). The accelerated adoption of cloud services has introduced complex challenges regarding the Shared Responsibility Model. To address this, an automated audit framework was developed using Infrastructure as Code (Terraform) and orchestration via CI/CD in GitHub Actions with parallelized matrix execution. This automated setup audited 15 distinct infrastructure scenarios against 165 security controls using the Checkov static analysis tool. The experimental results revealed a global failure rate of 60.6% in newly created resources, confirming the systemic insecurity of default configurations. The study demonstrates that cloud security requires a transition from manual checks to automated Guardrails in DevSecOps pipelines.*

Resumo. *Este trabalho apresenta uma análise comparativa das vulnerabilidades de segurança originadas por configurações padrão (default settings) nos principais provedores de nuvem (AWS, Azure e GCP). A adoção acelerada de serviços em nuvem introduziu desafios complexos no Modelo de Responsabilidade Compartilhada. Para investigar este cenário, desenvolveu-se um framework de auditoria automatizada utilizando Infraestrutura como Código (Terraform) e orquestração via CI/CD no GitHub Actions com estratégia de execução matricial paralela. Esta automação submeteu 15 cenários de infraestrutura distintos à auditoria de 165 controles de segurança via análise estática (Checkov). Os resultados revelaram uma taxa global de falha de 60,6% nos recursos recém-criados. O trabalho aponta que a segurança em nuvem exige a transição para Guardrails automatizados integrados às esteiras de desenvolvimento.*

1. Introdução

A maneira como as organizações adquirem e gerenciam sua infraestrutura de tecnologia da informação (TI) passou por uma mudança de paradigma fundamental na última década. O modelo tradicional de aquisição de infraestrutura de TI local (*on-premises*) vem sendo progressivamente substituído por um modelo de serviço, no qual os recursos computacionais são tratados como utilitários sob demanda [Mell and Grance 2011]. Essa transformação é impulsionada pela necessidade de agilidade e inovação das organizações. Neste cenário, o mercado de computação em nuvem, dominado por Amazon Web Services (AWS), Google Cloud Platform (GCP) e Microsoft Azure, consolidou-se como a espinha dorsal da economia digital. Dados recentes indicam que os gastos globais com serviços de nuvem alcançaram US$ 912,77 bilhões em 2025, com mais de 80% das empresas adotando estratégias multinuvem para evitar a dependência de um único provedor (*vendor lock-in*) [CloudZero 2025].

A migração acelerada da infraestrutura de TI das organizações para a nuvem, contudo, introduz novos e complexos vetores de ciber-riscos. A complexidade de proteger esses ambientes reside no Modelo de Responsabilidade Compartilhada. Enquanto o provedor garante a "segurança da nuvem" (física, rede global), o cliente deve garantir a "segurança na nuvem" [Amazon Web Services 2024]. A falha em compreender essa fronteira é a gênese das vulnerabilidades de configuração, acarretando consequências severas: o custo médio de uma violação de dados atingiu a marca de 4,6 milhões de dólares [IBM Security 2025]. Análises aprofundadas sobre a causa raiz desses ciberincidentes revelam um padrão: a principal ameaça frequentemente não reside em ataques externos de dia zero (*zero-day*), mas em falhas internas e erros humanos. Relatórios da Unit 42 indicam que mais de 70% dos ciberincidentes em nuvem são atribuídos a más configurações (*misconfigurations*) [Palo Alto Networks 2025].

Frequentemente, a origem dessas vulnerabilidades reside nas próprias configurações padrão (*default settings*) dos serviços oferecidos pelos provedores de nuvem. Para aumentar a facilidade de uso e a rapidez de implementação, os provedores muitas vezes entregam serviços com configurações permissivas. Tal prática desafia a implementação de arquiteturas de confiança zero (*zero trust*), uma vez que amplia a superfície de ataque inicial e dificulta a aplicação rigorosa do princípio do menor privilégio desde o provisionamento [Rose et al. 2020].

A principal contribuição deste trabalho é a análise sistemática da fragilidade das configurações padrão em serviços da AWS, Azure e GCP, desafiando a premissa de segurança nativa — definida neste estudo como o conjunto de controles e proteções ativados automaticamente pelo provedor no provisionamento do recurso, sem intervenções do usuário. Este diagnóstico é viabilizado pelo desenvolvimento de um framework de auditoria automatizada que valida, de forma experimental e reprodutível, a eficácia da detecção preventiva de riscos em esteiras de CI/CD.

Para tanto, implementa-se um *framework* de auditoria automatizada utilizando Infraestrutura como Código (*Infrastructure as Code* - IaC), com a ferramenta Terraform, e orquestração via CI/CD (*Continuous Integration/Continuous Delivery*), com a ferramenta GitHub Actions. O projeto é estruturado em um repositório unificado com isolamento estrito de estado, empregando estratégias de execução matricial para garantir a reprodutibilidade e a escalabilidade dos testes em múltiplos provedores simultaneamente. Esta

arquitetura permite auditar a postura de segurança intrínseca entregue nativamente pelos provedores, de forma sistemática e idêntica a um ambiente de produção real. A relevância desta avaliação reside em sua abordagem experimental ativa e na complexidade da engenharia de software aplicada, ao invés de basear-se apenas em revisões documentais. A aplicação desta metodologia revelou uma taxa de falha global de 60,6% nos recursos provisionados, confirmando a insegurança sistêmica das configurações padrão nos serviços ofertados pelos provedores.

O restante do artigo organiza-se da seguinte forma: a Seção 2 revisa trabalhos correlatos, seguida pelo detalhamento do *framework* na Seção 3. Os resultados quantitativos e sua análise qualitativa são apresentados, respectivamente, nas Seções 4 e 5. Por fim, a Seção 6 conclui o estudo.

2. Trabalhos Relacionados

A literatura sobre segurança em nuvem tem explorado o tema sob diversas óticas, desde a análise de *scripts* de automação até a análise comparativa de ferramentas, embora a análise específica de configurações padrão multinuvem via IaC apresente lacunas importantes.

Rahman e Rahman conduziram uma análise qualitativa pioneira sobre "*Security Smells*" em *scripts* de Infraestrutura como Código (IaC), escritos em Ansible e Chef. Os autores [Rahman and Rahman 2019] identificaram que a dependência implícita de configurações padrão atua como um anti-padrão de segurança, porque a facilidade de uso oculta dívidas técnicas de segurança que só se manifestam em produção.

No domínio das ferramentas de SAST (*Static Application Security Testing*) focadas em IaC, Pereira e Santos avaliam as mais utilizadas pela indústria (Checkov, KICS, Terrascan e tfsec) [Pereira and Santos 2022], destacando o Checkov pela alta cobertura de regras CIS (*Center for Internet Security*) [Center for Internet Security 2025]. Corroborando essa abordagem, a revisão sistemática de Kumara *et al.* reforça que a automação via SAST é o método mais eficaz para identificar vulnerabilidades antes do provisionamento dos recursos de infraestrutura [Kumara et al. 2021]. Contudo, Pereira e Santos alertam que o uso de ferramentas automatizadas não elimina a necessidade de triagem humana contextualizada para tratar falsos positivos.

Em outra vertente, Guptha *et al.* comparam os serviços de segurança nativos oferecidos pelos provedores de nuvem. O estudo, contudo, restringe-se ao levantamento qualitativo das funcionalidades, sem avaliar a eficácia da postura de segurança inicial (padrão) aplicada no momento da instanciação de recursos [Guptha et al. 2021]. Quanto à remediação, Khanam e Ahmad evidenciam a dificuldade de se aplicar técnicas de enrijecimento de sistemas (*hardening*) manualmente [Khanam and Ahmad 2020]. Complementarmente, Myrbakken e Colomo-Palacios [Myrbakken and Colomo-Palacios 2017] argumentam que ferramentas isoladas falham sem a adoção do paradigma *Shift-Left* — estratégia que antecipa as verificações de segurança para o início do ciclo de desenvolvimento de software.

O trabalho mais próximo ao escopo deste artigo é o de Verdet *et al.*, que apresenta um estudo empírico analisando a adoção de práticas de segurança em repositórios públicos do GitHub [Verdet et al. 2023]. No entanto, existe uma distinção metodológica crucial entre as abordagens.

Recentemente, Iosif *et al.* (2022) [Iosif et al. 2022] conduziram um estudo de larga escala sobre vulnerabilidades em *deploys* de nuvem, cujos resultados corroboram a persistência de falhas críticas de configuração mesmo em plataformas maduras. O presente trabalho expande essa discussão ao propor uma metodologia experimental multinuvem baseada em IaC, permitindo comparar como esses riscos se materializam de forma distinta entre os principais provedores do mercado.

Este artigo, diferente dos trabalhos citados, une a análise teórica de serviços oferecidos por diferentes provedores com uma validação experimental. O principal diferencial em relação ao estudo de Verdet *et al.* reside na abordagem: enquanto Verdet *et al.* se concetram na análise passiva de repositórios públicos para entender o comportamento do usuário (desenvolvedor), este artigo projeta e orquestra um fluxo de trabalho (*pipeline* de análise estática (SAST) em esteira de Integração Contínua, validando cenários de infraestrutura codificada para expor as vulnerabilidades nas configurações padrão de serviços entregues pelos provedores. A metodologia deste artigo isola, portanto, o risco de *design* da plataforma, um vetor frequentemente mascarado em análises de código de terceiros.

3. O *Framework* de Auditoria Automatizada

A metodologia fundamenta-se em uma abordagem híbrida, integrando pesquisa documental com validação experimental via IaC. O *framework* [Braga 2025] foi projetado para garantir a reprodutibilidade e o isolamento de variáveis, permitindo auditar a postura de segurança intrínseca dos provedores de forma sistemática. A automação é viabilizada por ferramentas de SAST para IaC, alinhadas à cultura DevSecOps e à filosofia *Shift Left* [Myrbakken and Colomo-Palacios 2017].

Para a seleção do ferramental, utiliza-se o Terraform por sua natureza declarativa e agnóstica, permitindo codificar cenários consistentes para AWS, Azure e GCP [Braga 2025]. A análise de segurança é centrada no Checkov, selecionado por sua capacidade de análise baseada em grafos, que identifica vulnerabilidades relacionais superiores a ferramentas baseadas estritamente em expressões regulares (*Regex*) [Pereira and Santos 2022]. Todo o processo é orquestrado via GitHub Actions, embora a lógica de auditoria seja portável para outros ambientes de CI/CD (como GitLab CI ou Jenkins), garantindo a flexibilidade da solução.

O repositório [Braga 2025] adota uma organização hierárquica rigorosa em três níveis para isolar os ambientes de teste: (i) diretórios raiz por provedor; (ii) subdivisão pelos cinco serviços críticos (*Core Services*); e (iii) projetos Terraform independentes. A delimitação do escopo para este conjunto fundamenta-se no Princípio de Pareto, focando nas categorias estruturais que compõem a espinha dorsal das arquiteturas modernas e concentram a maior superfície de ataque corporativa [Gartner 2025, Center for Internet Security 2025]. A Tabela 1 sintetiza a matriz de 15 cenários auditados.

O cerne do experimento reside na omissão intencional de argumentos nos arquivos de configuração, forçando a aplicação das configurações padrão [Center for Internet Security 2025]. Esta abordagem metodológica visa isolar o risco de *design* da plataforma, um vetor frequentemente mascarado em análises de código de terceiros.

A inclusão de novos provedores ao framework, como Oracle Cloud ou Huawei Cloud, é facilitada pela arquitetura modular adotada. O processo exige apenas a criação

Tabela 1. Matriz de Cenários de Infraestrutura Auditados por Provedor.

Categoria	AWS	Azure	GCP
Computação	EC2 Instance	Windows VM	Compute Instance
Armazenamento	S3 Bucket	Storage Account	Cloud Storage
Banco de Dados	RDS Instance	MSSQL Server	Cloud SQL
Identidade	IAM Role/User	IAM Member	IAM Service Account
Kubernetes	EKS Cluster	AKS Cluster	GKE Cluster

de um novo diretório raiz e a definição dos recursos equivalentes utilizando os respectivos provedores do Terraform, mantendo a consistência da análise estática via Checkov.

3.1. Divergências de Configuração e Modelos de Segurança

A análise revelou que a verbosidade necessária para atingir o estado mínimo de provisionamento não é uniforme entre os provedores, expondo modelos de segurança distintos:

- AWS e GCP (Modelos Minimalistas): Nestes provedores, a omissão de parâmetros resulta na aceitação implícita de recursos padrão permissivos [Rose et al. 2020, OWASP Foundation 2023]. Na AWS, isso implica no uso da *Default VPC* e do IMDSv1 [Center for Internet Security 2025]. No GCP, a omissão de conta de serviço concede permissões excessivas à instância, enquanto o uso da rede *default* expõe o recurso a regras de *firewall* vulneráveis [Center for Internet Security 2025]. Tais práticas fragilizam a implementação de arquiteturas *zero trust*, pois assumem a confiabilidade de redes e identidades desde o provisionamento [Rose et al. 2020, OWASP Foundation 2023].
- Microsoft Azure (Modelo Verboso): O Azure exige uma declaração explícita de recursos auxiliares como VNet e NIC [Braga 2025]. Contudo, a vulnerabilidade manifesta-se pela omissão de recursos de segurança inteiros: o provedor permite instanciar VMs sem a associação de um *Network Security Group* (NSG), resultando em ativos sem qualquer filtragem de tráfego nativa [Center for Internet Security 2025].

O *framework* utiliza a estratégia de execução matricial (`strategy.matrix`) para processar os 15 cenários em paralelo [Braga 2025]. O Checkov opera em modo não-bloqueante (`soft_fail: true`), gerando artefatos em SARIF e JSON que detalham desde o *ID* do controle violado até o impacto potencial, provendo a base para o cálculo da Taxa de Falha Global apresentada na Seção 4 [Braga 2025].

4. Resultados

Esta seção apresenta os dados brutos e estatísticos obtidos através da execução do *framework*. Os resultados detalham o desempenho da ferramenta de análise estática frente aos 15 cenários de infraestrutura codificados.

Em uma visão geral quantitativa, o *framework* processou com sucesso todos os arquivos de definição de infraestrutura, executando um total de 165 verificações de

segurança distintas (*checks*) baseadas no CIS Benchmark [Center for Internet Security 2025]. A análise identificou um total de 100 vulnerabilidades de configuração contra apenas 65 aprovações. Objetivando normalizar as discrepâncias de granularidade entre provedores, calcula-se a Taxa de Falha Global (R_{fail}) conforme a Equação 1. Considera-se $n = 15$ o total de cenários, onde $V_{detected,i}$ e $C_{total,i}$ representam, respectivamente, as falhas identificadas e os controles aplicados ao i-ésimo cenário.

$$R_{fail} = \left(\frac{\sum_{i=1}^{n} V_{detected,i}}{\sum_{i=1}^{n} C_{total,i}} \right) \times 100 \quad (1)$$

O índice global obtido foi de 60,6%. Adicionalmente, avaliou-se a densidade de falhas considerando os 26 recursos efetivamente provisionados (incluindo serviços principais e dependências auxiliares). A razão entre o total de falhas identificadas (100) e recursos criados resultou em uma média de 3,8 vulnerabilidades por componente, indicando que a configuração padrão introduz quase quatro vetores de risco por ativo instanciado. A Tabela 2 consolida os resultados por provedor.

Tabela 2. Resumo Geral da Análise de Segurança por Provedor.

Provedor	**Total de *Checks***	**Aprovados**	**Reprovados**	**Taxa de Falha**
AWS	47	20	27	57,4%
Azure	59	20	39	66,1%
GCP	59	25	34	57,6%
Total	165	65	100	60,6%

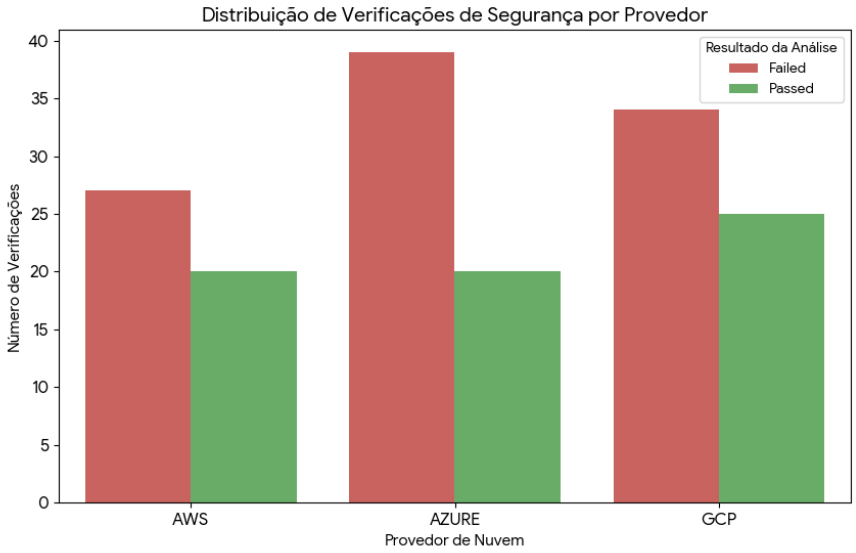

Figura 1. Distribuição de Verificações de Segurança (Aprovadas vs. Reprovadas).

Os dados evidenciam uma disparidade onde o Microsoft Azure (66,1%) supera as taxas de AWS e GCP (próximas a 57%), sugerindo uma correlação direta entre a verbo-

sidade de provisionamento e a densidade de falhas detectadas. A Figura 1 corrobora essa análise ao demonstrar a predominância absoluta de reprovações em todos os cenários, confirmando a insegurança das configurações padrão como um fenômeno sistêmico e transversal, e não uma deficiência isolada de um provedor.

Avançando na granularidade da análise para além da visão consolidada por provedor, buscou-se identificar os vetores de risco mais agudos em nível de recurso. Conforme detalhado na Tabela 3, os serviços de orquestração de contêineres e bancos de dados gerenciados lideram o ranking de "Top Ofensores". Um destaque negativo é o recurso `azurerm_mssql_server`, banco de dados gerenciado (PaaS) que atingiu 100% de reprovação. Este resultado evidencia que serviços do tipo PaaS tendem a apresentar configurações padrão significativamente mais vulneráveis do que a infraestrutura básica.

Tabela 3. Ranking de Recursos Críticos (Top Offenders).

Recurso (Terraform)	Provedor	Falhas	Taxa de Falha
azurerm_kubernetes_cluster	Azure	15	75,0%
azurerm_storage_account	Azure	11	73,3%
aws_s3_bucket	AWS	7	63,6%
google_container_cluster	GCP	12	63,1%
aws_db_instance	AWS	10	62,5%
azurerm_mssql_server	Azure	7	100,0%

Além dos valores absolutos, a análise qualitativa das reprovações permitiu isolar falhas técnicas específicas de alta severidade. A Tabela 4 detalha uma amostragem dessas vulnerabilidades, correlacionando o serviço com a descrição precisa do risco.

Tabela 4. Amostragem de falhas de alta severidade detectadas.

Serviço	ID da Verificação	Descrição do Risco
EC2 (AWS)	CKV_AWS_79	Metadados de instância (IMDSv2) não exigidos.
S3 (AWS)	CKV_AWS_18	Logging de acesso ao servidor desativado.
VM (Azure)	CKV_AZURE_1	Autenticação SSH por senha permitida.
AKS (Azure)	CKV_AZURE_4	Monitoramento do Cluster desativado.
GCE (GCP)	CKV_GCP_30	OS Login desativado.
GKE (GCP)	CKV_GCP_66	Uso de Basic Auth (Legado).

Observa-se que falhas críticas, como autenticação fraca (SSH via senha) e falta de proteção em metadados (IMDSv2), confirmam que os recursos foram instanciados em violação às políticas de segurança do *benchmark* CIS [Center for Internet Security 2025]. Esses dados evidenciam que a postura inicial da infraestrutura ("Day 0") é inerentemente frágil, exigindo que o engenheiro atue proativamente na refatoração do código para fechar vetores de ataque que, por padrão, nascem abertos.

Ao segmentar a análise por categoria de serviço, é possível mapear a densidade de falhas em diferentes camadas da infraestrutura. A Figura 2 apresenta o mapa de calor das vulnerabilidades detectadas.

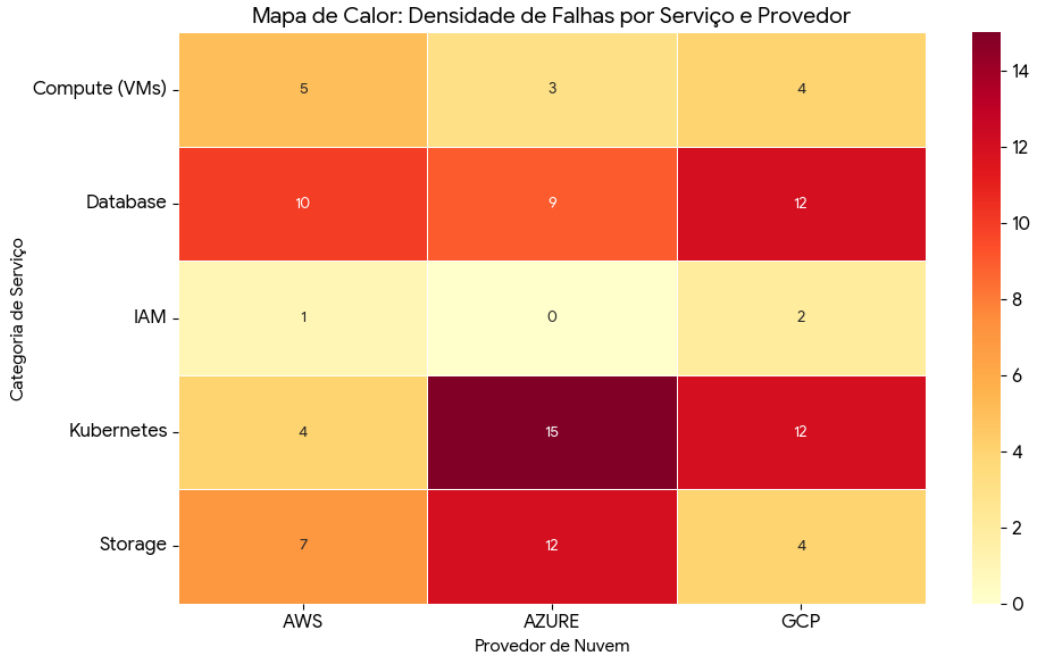

Figura 2. Mapa de Calor: Densidade de Falhas por Serviço e Provedor.

No escopo deste estudo experimental,os dados confirmam que serviços de maior abstração (PaaS), como Kubernetes e Bancos de Dados, são provisionados com configurações padrão que possuem mais vulnerabilidades do que serviços básicos de IaaS, como máquinas virtuais.

A aparente disparidade de dados entre o *ranking* da Tabela 3 e o mapa de calor da Figura 2 deve-se a recursos auxiliares de auditoria e criptografia que, somados ao recurso principal, elevam a densidade de vulnerabilidades da categoria. Enquanto a tabela foca em recursos individuais, como o `azurerm_mssql_server`, o mapa de calor agrega todas as falhas da categoria de serviço, como a categoria *"Database"*.

Conclui-se a apresentação dos resultados com a validação da execução experimental. A etapa de análise estática processou integralmente os 15 projetos de infraestrutura definidos na metodologia. A execução automatizada no pipeline de CI/CD ocorreu sem interrupções ou erros de interpretação sintática, garantindo a integridade da base de dados utilizada.

5. Discussão

Esta seção interpreta os resultados apresentados anteriormente, estabelecendo relações entre os dados quantitativos obtidos a partir da execução do *framework*, as arquiteturas de segurança dos provedores de nuvem e a literatura agnóstica de provedor.

Ressalta-se que a relevância dos achados sobrepuja a métrica absoluta de cenários testados. Ao auditar os pilares fundamentais da nuvem (IaaS, PaaS e K8s), o estudo captura a postura de segurança dos recursos que efetivamente sustentam as cargas de trabalho de produção. As falhas detectadas, portanto, não representam casos de borda em serviços periféricos, mas indicam uma fragilidade sistêmica nos componentes onipresentes no mercado.

Uma análise detalhada das vulnerabilidades transversais demonstra que falhas idênticas detectadas em múltiplos provedores confirmam riscos arquiteturais que vão além da implementação técnica. A Tabela 5 correlaciona esses vetores de risco comuns com exemplos reais detectados pelo Checkov e seu respectivo impacto de negócio, evidenciando consequências como ataques remotos e impossibilidade de forense digital.

Tabela 5. Análise Qualitativa: Riscos Transversais e Impacto.

Vetor de Risco	**Exemplo (ID Checkov)**	**Impacto de Negócio**
Exposição de Rede	CKV_AWS_39 (EKS Public API)	Ataque Remoto: API de gerenciamento exposta à internet pública.
Falta de Logs	CKV_AZURE_23 (SQL Auditing)	Cegueira: Impossibilidade de forense digital e violação de auditoria.
Criptografia	CKV_GCP_38 (CSEK Missing)	Soberania: Dependência do provedor para destruição segura (*Crypto-shredding*).

A exposição inadvertida de recursos à Internet pública foi detectada consistentemente nos três provedores, violando controles básicos de segmentação. O contexto de ameaça é crítico: a exposição da API do Kubernetes remove a barreira de rede que mitigaria vulnerabilidades como a CVE-2018-1002105 [NIST National Vulnerability Database 2018].

Adicionalmente, identificou-se um risco de governança em criptografia. A predominância de chaves geridas pela plataforma cria um conflito com requisitos de soberania de dados. A ausência de CMEK (*Customer-Managed Encryption Keys"*) impede a execução do *crypto-shredding* (destruição de dados via exclusão da chave), técnica essencial para o "Direito ao Esquecimento"da GDPR. Outro ponto crítico é a cegueira em auditoria (Logging). A desabilitação de logs por padrão é a falha de observabilidade mais crítica, cujo contexto de ameaça remete ao padrão de ataque CAPEC-93 (Log Injection-Tampering-Forging) [MITRE Corporation 2024]. Sem uma trilha de auditoria imutável, um atacante pode operar no ambiente por meses sem gerar alertas.

As vulnerabilidades transversais identificadas não representam apenas riscos teóricos, mas desvios mensuráveis que podem ser categorizados sistematicamente. Para qualificar o impacto destas falhas, utilizou-se o modelo de ameaças STRIDE (*Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege*) [Shostack 2014]. Esta análise permite transpor as falhas técnicas detectadas pelo Checkov para vetores de ataque reais que comprometem as propriedades de segurança fundamentais do ambiente de rede e os princípios de arquitetura *Zero Trust* [Rose et al. 2020], conforme detalhado na Tabela 6, que correlaciona a propriedade de segurança violada com o achado técnico.

Ao aprofundar a análise desses vetores é possível detectar uma fragilidade sistêmica que impacta diretamente a resiliência da infraestrutura. No domínio do *Spoofing*, a persistência de métodos de autenticação fraca, como o IMDSv1 na AWS (detectado via CKV_AWS_79), facilita ataques de *Server-Side Request Forgery* (SSRF), permitindo que um atacante usurpe a identidade da instância para extrair credenciais temporárias da *IAM Role*.

Tabela 6. Aplicação do STRIDE: Propriedades e Riscos Identificados.

Ameaça	**Propriedade Violada**	**Materialização na Pesquisa (Exemplos)**
Spoofing	Autenticidade	Uso de autenticação legada (*Basic Auth*) no GKE ou SSH via senha.
Tampering	Integridade	Ausência de proteção contra exclusão (*MFA Delete*) em logs de auditoria.
Repudiation	Não-Repúdio	Logs desativados por padrão (RDS e Cloud SQL), impedindo rastreabilidade.
Info. Disclosure	Confidencialidade	Discos não criptografados ou APIs de gerenciamento (Kubernetes) expostas.
Denial of Service	Disponibilidade	Ausência de cotas de recursos, permitindo exaustão de CPU/Memória.
Elevation	Autorização	*Service Accounts* padrão com permissões excessivas (*Over-privileged*).

Quanto ao *Tampering* e *Repudiation*, a desativação por padrão de logs de auditoria e a ausência de travas de exclusão (*MFA Delete*) criam um cenário de "cegueira forense". Sem trilhas de auditoria imutáveis, torna-se impossível garantir o não-repúdio, permitindo que exfiltrações de dados ocorram sem detecção ou rastreabilidade histórica, violando requisitos mandatórios da LGPD e normas de *compliance*.

A ameaça de *Information Disclosure* manifesta-se criticamente na exposição de APIs de gerenciamento de *clusters* (K8s). O acesso público por padrão remove a barreira de rede que mitigaria a exploração de vulnerabilidades de dia zero na API do Kubernetes. Complementarmente, o *Denial of Service* (DoS) manifesta-se como um impacto econômico na nuvem: a ausência de limites de recursos em contêineres pode ser explorada para causar um *Economic Denial of Sustainability* (EDoS), onde o consumo desenfreado de recursos infla os custos operacionais até a exaustão financeira do cliente.

Além disso, a Elevação de Privilégio (*Elevation*) é potencializada pelo uso de contas de serviço com permissões excessivas. No modelo de responsabilidade compartilhada, essa configuração padrão expande o alcance e a criticidade de um incidente: o comprometimento de uma aplicação periférica concede ao atacante poderes de administrador sobre o plano de controle da nuvem, permitindo o movimento lateral e a persistência no ambiente.

Os resultados validam também o "Paradoxo da Complexidade em PaaS". Observou-se que quanto maior o nível de gerenciamento e abstração oferecido pelo provedor, mais permissiva tende a ser a configuração padrão. Os clusters Kubernetes lideraram o volume de falhas em todos os provedores. Por sua natureza complexa, esses serviços são entregues com APIs públicas e autenticação legada habilitadas para reduzir a curva de aprendizado inicial. Para equipes de segurança, isso inverte a lógica tradicional: serviços gerenciados exigem, na prática, um esforço de *hardening* inicial (via código) superior ao de servidores tradicionais para atingir *compliance*.

Outra constatação do estudo foi a dicotomia entre verbosidade e abstração na

comparação Azure vs AWS/GCP. A taxa de falha superior do Microsoft Azure (66,1%) revela uma causa raiz arquitetural: a diferença entre modelos de configuração explícitos e implícitos. O provedor Azure adota um modelo de alta verbosidade e desacoplamento. Para provisionar uma máquina virtual segura, o engenheiro deve declarar explicitamente recursos auxiliares, como o NSG. Se o engenheiro omite esse recurso, a ferramenta de análise estática detecta imediatamente a ausência de proteção. Em contraste, AWS e GCP adotam um modelo de abstração implícita. Ao omitir configurações de rede, esses provedores preenchem as lacunas silenciosamente com recursos padrão. Paradoxalmente, a verbosidade do Azure favorece a detectabilidade, enquanto a abstração da AWS/GCP pode mascarar riscos sistêmicos.

A análise experimental também revela uma divergência sistemática entre as recomendações de segurança publicadas pelos provedores e as configurações aplicadas nativamente no provisionamento. Embora os provedores disponibilizem guias de boas práticas, como o *AWS Well-Architected Framework* [AMAZON WEB SERVICES 2024], o *Microsoft Cloud Adoption Framework* [MICROSOFT AZURE 2024] e o *Google Cloud Security Foundations* [GOOGLE CLOUD 2025], o estado inicial dos recursos prioriza a disponibilidade funcional em detrimento dos controles de segurança especificados nestes mesmos documentos.

No caso da AWS, o pilar de segurança do seu *framework* de arquitetura [AMAZON WEB SERVICES 2024] recomenda a aplicação rigorosa do Princípio do Menor Privilégio e a proteção de metadados. Contudo, a persistência do protocolo IMDSv1 como padrão (detectada pela falha CKV_AWS_79) contraria as diretrizes de mitigação de ataques SSRF (*Server-Side Request Forgery*) documentadas pelo próprio provedor. Factualmente, o recurso é entregue com uma superfície de ataque que o provedor instrui, em seus manuais de enrijecimento (*hardening*), a desativar.

A inconsistência é igualmente observada no Google Cloud Platform. As diretrizes do *Google Cloud Security Foundations* [GOOGLE CLOUD 2025] preconizam o uso de identidades granulares; todavia, a configuração padrão associa instâncias de computação à Conta de Serviço Padrão com o papel de Editor. Esta atribuição concede permissões de escrita e modificação em quase todos os recursos do projeto, violando a recomendação de granularidade de acesso presente no guia de arquitetura oficial.

Identifica-se, portanto, um *Compliance Gap* intrínseco ao provisionamento. No Modelo de Responsabilidade Compartilhada [AMAZON WEB SERVICES 2024], o provedor garante a segurança da infraestrutura global, mas entrega o recurso individual em um estado que exige refatoração imediata para atingir os níveis de conformidade que o próprio provedor define como ideais. Essa disparidade evidencia que a segurança nativa não é uma propriedade passiva da plataforma, mas uma variável dependente de intervenção ativa via Infraestrutura como Código.

Uma vez categorizados os riscos, torna-se necessário confrontar a realidade da configuração padrão com a arquitetura de segurança recomendada. A Tabela 7 sintetiza essa dicotomia, demonstrando como a aplicação de controles compensatórios (*Hardening*) neutraliza os vetores de ataque identificados na análise anterior.

A discrepância evidenciada na Tabela 7 reforça as implicações para a cultura DevSecOps. A média de 3,8 vulnerabilidades por recurso torna insustentável a dependência

Tabela 7. Comparativo: Default vs. Hardened sob a ótica de Ameaças

Vetor	Configuração Padrão	Mitigação (Hardened)
Rede	Exposta. Facilita exploração remota.	Privada. Quebra a *Kill Chain*.
Chaves	Geridas pelo Provedor. Risco de acesso.	Geridas pelo Cliente (CMEK).
Logs	Desabilitados. Cegueira operacional.	Habilitados. Forense digital.

da verificação humana, validando a necessidade urgente de transição de um modelo de verificações manuais pontuais para controles preventivos automatizados via SAST. Em última instância, o cenário de insegurança sistêmica (60,6% de reprovação) refuta a premissa de proteção nativa: na era da infraestrutura programável, a segurança não é um *commodity* da plataforma, mas uma disciplina de engenharia que deve ser explicitamente codificada (*Secure by Design*).

6. Conclusão

Este artigo confirmou empiricamente que a premissa de "Segurança na Nuvem" não pode se apoiar nos padrões de fábrica dos provedores. A identificação de uma taxa global de falhas de 60,6% nos recursos recém-criados demonstra que a insegurança é uma característica sistêmica do provisionamento padrão. A análise revelou que os provedores priorizam a usabilidade e a redução de atrito em detrimento do princípio *Secure by Default*, transferindo uma carga de configuração excessiva para o cliente.

A pesquisa confirmou que a segurança em nuvem não é um atributo intrínseco da plataforma, exigindo uma mudança de paradigma. Com uma densidade média de 3,8 vulnerabilidades por recurso, demonstrou-se a inviabilidade da auditoria manual e a necessidade imperativa de codificar a segurança como requisito de engenharia, utilizando automação como um caminho sustentável para ambientes escaláveis.

É importante ressaltar que a abordagem baseada em análise estática (SAST) limita-se à validação do código antes do provisionamento ("Day 0/1"). Adicionalmente, o estudo reconhece como limitação o uso exclusivo da ferramenta Checkov. Embora seja uma referência de mercado pela sua análise baseada em grafos, o emprego de múltiplos motores de auditoria (como o *tfsec* e o KICS) em trabalhos futuros poderia refinar a detecção, reduzindo possíveis falsos negativos e ampliando a cobertura de regras específicas. O estudo também não abrange a detecção de derivação de configuração (*Configuration Drift*) que possa ocorrer em tempo de execução ("Day 2") devido a intervenções manuais ou alterações não gerenciadas.

Como recomendações estratégicas diante dos riscos sistêmicos, sugere-se a adoção de módulos "Golden Image", onde engenheiros de plataforma criam bibliotecas que encapsulem configurações seguras (ex: CMEK e *logs* forçados), expondo apenas parâmetros seguros aos desenvolvedores. Além disso, recomenda-se a implementação de *Guardrails* automatizados, onde a análise estática (SAST) atua como um *Quality Gate* bloqueante no pipeline de CI/CD, rejeitando automaticamente deploys com configurações inseguras transversais.

Para trabalhos futuros, sugere-se investigar a implementação de bots de auto-

remediação (*Self-Healing*) que corrijam derivações de configuração em tempo real, a realização de estudos sobre o custo financeiro (*Security Debt*) necessário para adequar ambientes legados aos *benchmarks* de segurança, bem como a extensão da metodologia para análise dinâmica (DAST) em tempo de execução.

Agradecimentos

Este trabalho é parcialmente financiado pelo CNPq, CAPES, FAPERJ e RNP e faz parte do INCT de Redes de Comunicação e Internet das Coisas Inteligentes (ICoNIoT), financiado pelo CNPq (405940/2022-0) e pela CAPES (88887.954253/2024-00).

Os autores usaram a ferramenta de inteligência artifical generativa Gemini exclusivamente para correção ortográfica e gramatical do texto e auxílio para geração de figuras, não substituindo a análise crítica dos resultados pelos autores.

Referências

Amazon Web Services (2024). Aws shared responsibility model. https://aws.amazon.com/compliance/shared-responsibility-model/. Acesso em: 7 out. 2025.

AMAZON WEB SERVICES (2024). Aws well-architected framework: Security pillar. https://docs.aws.amazon.com/wellarchitected/latest/security-pillar/welcome.html. Acesso em: 01 abr. 2026.

Braga, C. (2025). Repositório do Framework: Análise de Configurações Padrão em Nuvem. https://github.com/CarolineBraga/tcc-cloud-defaults-security. Acesso em: 28 mar. 2026.

Center for Internet Security (2025). Cis benchmarks. Acesso em: 7 out. 2025.

CloudZero (2025). 90+ estatísticas de computação em nuvem: um instantâneo do mercado de 2025. https://www.cloudzero.com/blog/cloud-computing-statistics/. Acesso em: 27 out. 2025.

Gartner (2025). Forecast: Public cloud services, worldwide, 2023-2027, 4q24 update. Technical report, Gartner Research.

GOOGLE CLOUD (2025). Google cloud architecture framework: Security, privacy, and compliance. https://cloud.google.com/architecture/framework/security. Acesso em: 01 abr. 2026.

Guptha, A. S., Murali, H., and T., S. (2021). A comparative analysis of security services in major cloud service providers. In *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 1533–1538. IEEE.

IBM Security (2025). Cost of a data breach report 2025. Technical report, Ponemon Institute. Acesso em: 28 set. 2025.

Iosif, A.-C., Gasiba, T. E., Zhao, T., Lechner, U., and Pinto-Albuquerque, M. (2022). A large-scale study on the security vulnerabilities of cloud deployments. In *Ubiquitous Security: First International Conference, UbiSec 2021*, pages 37–52. Springer.

Khanam, S. and Ahmad, M. (2020). Cloud security hardening using cis benchmarks: A case study. In *2020 International Conference on Computing and Information Technology*. IEEE.

Kumara, I., Pietrantuono, R., and Russo, S. (2021). Security of infrastructure as code: A systematic literature review. *IEEE Access*, 9:109641–109667.

Mell, P. and Grance, T. (2011). The nist definition of cloud computing. Technical Report SP 800-145, National Institute of Standards and Technology, Gaithersburg, MD.

MICROSOFT AZURE (2024). Azure security benchmark v3: Network security. https://learn.microsoft.com/en-us/security/benchmark/azure/security-controls-v3-network-security. Acesso em: 01 abr. 2026.

MITRE Corporation (2024). Capec-93: Log injection-tampering-forging. https://capec.mitre.org/data/definitions/93.html. Acesso em: 22 nov. 2025.

Myrbakken, H. and Colomo-Palacios, R. (2017). Devsecops: A multivocal literature review. In *Software Process Improvement and Capability Determination (SPICE 2017)*, pages 17–29. Springer.

NIST National Vulnerability Database (2018). Cve-2018-1002105: Kubernetes privilege escalation. https://nvd.nist.gov/vuln/detail/CVE-2018-1002105. Acesso em: 22 nov. 2025.

OWASP Foundation (2023). Owasp cloud native security top 10. https://owasp.org/www-project-cloud-native-application-security-top-10/. Acesso em: 8 out. 2025.

Palo Alto Networks (2025). Unit 42 cloud threat report, 2h 2025. https://www.paloaltonetworks.com/prisma/unit42-cloud-threat-research. Acesso em: 28 set. 2025.

Pereira, M. and Santos, N. (2022). Evaluation of static application security testing tools for infrastructure as code. In *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 167–174. IEEE.

Rahman, A. and Rahman, M. R. (2019). Security smells in ansible and chef scripts: A qualitative study and a dataset. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, pages 430–441. IEEE.

Rose, S., Souppaya, M., Fagan, M., and Frankel, K. (2020). Zero trust architecture. Technical Report SP 800-207, National Institute of Standards and Technology, Gaithersburg, MD.

Shostack, A. (2014). *Threat Modeling: Designing for Security*. Wiley, Indianapolis, IN.

Verdet, A., Hamdaqa, M., Da Silva, L., and Khomh, F. (2023). Exploring security practices in infrastructure as code: An empirical study. *IEEE Transactions on Software Engineering*, 49(10):4620–4645.