UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ANDERSON ROBERTO DEIZEPE

DETECÇÃO DE *SMELLS* EM POLÍTICAS DE SEGURANÇA PARA PLATAFORMAS CLOUD

DISSERTAÇÃO DE MESTRADO

CORNÉLIO PROCÓPIO

2023

ANDERSON ROBERTO DEIZEPE

DETECÇÃO DE *SMELLS* EM POLÍTICAS DE SEGURANÇA PARA PLATAFORMAS CLOUD

Detection of Smells in Security Policies for Cloud Platforms

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Mestre em Informática.

Orientadora: Prof^a. Dr^a. Katia Romero Felizardo Scannavino

Co-orientador: Prof. Dr. André Takeshi Endo

CORNÉLIO PROCÓPIO
2023

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ANDERSON ROBERTO DEIZEPE

DETECÇÃO DE SMELLS EM POLÍTICAS DE SEGURANÇA PARA PLATAFORMAS CLOUD

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Informática da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Computação Aplicada.

Data de aprovação: 15 de Agosto de 2023

Dra. Katia Romero Felizardo Scannavino, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Erica Ferreira De Souza, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Roberto Pereira, Doutorado - Universidade Federal do Paraná (Ufpr)

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 15/08/2023.

Dedico esse trabalho aos meus familiares, amigos e todas as pessoas de boa vontade que valorizam a educação.

AGRADECIMENTOS

À minha companheira e esposa Jéssica, pelo apoio, paciência, incentivo e compreensão e aos meus pais, Elisabete e Roberto, por participarem da minha formação como pessoa.

Agradeço a minha orientadora, Prof^{a}. Dr^{a}. Katia Romero Felizardo Scannavino, pela compreensão e apoio e ao Prof Dr. André Takeshi Endo, pela organização, excelência e incentivo.

À instituição UTFPR, pelos recursos humanos e materiais, excelência e qualidade no ensino e ao secretario do programa, pelo apoio e assistência, sempre de pronto.

Aos colegas conterrâneos que viveram a experiência do programa: Alex, André, Eduardo, Glauco, Luís e Marcos, companheiros de viagens, pelos bons e maus momentos vividos e pela ajuda.

Aos professores e amigos do IFSP, campus de Presidente Epitácio - SP e Caraguatatuba - SP, pelo incentivo e motivação.

Aos meus amigos Antônio Jr. (*in memoriam*), Giovanni (*in memoriam*), Lucas, Nikolas, João, Carmem, Mari, Lu, Carol, Érica, Júlio, Natália, Giovani, Vilson, André, Silas, que me ouviram, apoiaram e incentivaram minha jornada na educação.

À vó Cirlei, Ivoni, Nena e Vaidi, pela motivação e apoio quando eu mais precisei, incentivo e lição de vida.

A todos os meus professores, desde a educação básica, graduação, pós-graduação e cursos que fizeram parte da minha educação.

Aos meus alunos e orientandos, que confiam no meu trabalho em contribuir humildemente na sua formação.

Ao Prof. Dr. Robson Quintilio, por me inspirar na carreira acadêmica e ser meu exemplo de exercício de docência.

A todos que contribuíram direta ou indiretamente com este trabalho.

“Faça o teu melhor, na condição que você tem, enquanto você não tem condições melhores para fazer melhor ainda!”

Mario Sergio Cortella

RESUMO

DEIZEPE, Anderson Roberto. DETECÇÃO DE *SMELLS* EM POLÍTICAS DE SEGURANÇA PARA PLATAFORMAS CLOUD. 67 f. Dissertação – Programa de Pós-Graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2023.

Contexto: Com o crescente aumento de demanda por computação em nuvem, a preocupação com segurança é cada vez mais presente. A fim de centralizar o gerenciamento de acesso e segurança, provedores de computação em nuvem implementam o modelo *Security-as-a-service* (Segurança como serviço), que representa a segurança oferecida como um serviço em nuvem. Um *code smell* é um sinal que normalmente corresponde a um problema no sistema, embora não necessariamente. Em casos de privilégios excedentes, a aplicação pode estar exposta, com riscos de segurança e, caso o usuário tenha menos privilégios que os necessários, a aplicação apresentará um mau funcionamento. Detectar *smells* em políticas de segurança para plataformas *cloud* implica em verificar se existem excessos ou ausência de privilégios ao usuário. **Objetivo:** O objetivo deste trabalho foi desenvolver uma ferramenta de análise estática de código que identifica *smells* de segurança em políticas de segurança para aplicações na nuvem. Com a ferramenta foi possível mapear permissões necessárias e comparar com permissões concedidas ao usuário do IAM programático, que é o usuário atribuído a aplicação no ambiente de produção, para que tenha acesso aos recursos e serviços *cloud*; fornecendo um relatório de excesso e/ou falta de privilégios. Para este trabalho foi escolhido o provedor Amazon AWS, por ser a plataforma mais amplamente utilizada. **Método:** Inicialmente, foi realizado um levantamento de *smells* em respostas a perguntas referentes a permissões de usuários programáticos em ambiente *cloud*, no fórum StackOverflow. Posteriormente, um mapeamento foi conduzido com o intuito de identificar métodos e APIs do SDK da AWS que se relacionam com políticas de segurança e acesso. Com o apoio deste mapeamento, uma ferramenta foi desenvolvida e, com informações fornecidas pelo desenvolvedor, produziu relatórios com *smells* de segurança. Por fim, foi conduzida uma avaliação experimental e a abordagem e ferramenta foram avaliadas em aplicações de código livre. **Conclusões:** A abordagem proposta mostrou-se promissora, sendo capaz de identificar *smells* tanto em projetos de código aberto, via análise de código fonte, quanto no usuário programático IAM da nuvem disponibilizado para testes. Notou-se uma assertividade superior a 90% e tempo de execução dos testes em aplicações complexas satisfatórios, inferiores a 185 segundos. Este trabalho apresenta contribuições na área de segurança e implementação de aplicações web em provedores de nuvem, contribui com a identificação de *smells*.

Palavras-chave: Computação em nuvem, segurança, permissões, smells, políticas de segurança, IAM, AWS, análise estática

ABSTRACT

DEIZEPE, Anderson Roberto. Detection of Smells in Security Policies for Cloud Platforms. 67 f. Dissertação – Programa de Pós-Graduação em Informática, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2023.

Background: With the increasing demand for cloud computing, security concerns are increasingly present. In order to centralize access and security management, cloud computing providers implement the Security-as-a-service model, which represents security offered as a cloud service. A code smell is a signal that normally corresponds to a problem in the system, although not necessarily. In cases of excess privileges, the application may be exposed, with security risks and, if the user has fewer privileges than necessary, the application will malfunction. Detecting smells in security policies for cloud platforms implies checking whether there are excesses or absences of user privileges. **Objective:** The objective of this work was to develop a static code analysis tool that identifies security smells in security policies for cloud applications. With the tool, it was possible to map necessary permissions and compare with permissions granted to the programmatic IAM user, which is the user assigned to the application in the production environment, so that it has access to cloud resources and services; providing a report of excess and/or lack of privileges. For this work, the Amazon AWS provider was chosen, as it is the most widely used platform. **Method:** Initially, a survey of smells was carried out in answers to questions regarding permissions of programmatic users in a cloud environment, in the StackOverflow forum. Subsequently, a mapping was conducted in order to identify AWS SDK methods and APIs that relate to security and access policies. With the support of this mapping, a tool was developed and, with information provided by the developer, produced reports with security smells. Finally, an experimental evaluation was conducted and the approach and tool were evaluated in open source applications. **Conclusions:** The proposed approach proved to be promising, being able to identify smells both in open source projects, via source code analysis, and in the programmatic IAM user of the cloud available for testing. It was noted an assertiveness superior to 90% and satisfactory execution time of the tests in complex applications, inferior to 185 seconds. This work presents contributions in the area of security and implementation of web applications in cloud providers, contributes to the identification of smells.

Keywords: Cloud computing, security, permissions, smell, security policies, IAM, AWS, static analysis

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

IAM	Identity and Access Management
DDoS	Distributed Denial of Service
PHP	PHP: Hypertext Preprocessor
GPL	General Public License
SDK	Software development kit
AWS	Amazon Web Services
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
GCP	Google Cloud Platform
AWS	Amazon Web Services
S3	Simple Storage Service
ARN	Amazon Resource Name
SES	Simple Email Service
SQS	Simple Queue Service
FIFO	First-in, First-out
SNS	Simple Notification Service
A2A	Application to Application
A2P	Application to Pearson
SMS	Short Message Service
IAM	Identity and Access Management
VPS	Virtual Private Server

SUMÁRIO

1 INTRODUÇÃO

Computação em nuvem (do inglês: *Cloud Computing*) tem ganhado cada vez mais espaço e se tornado um modelo de computação distribuída bastante popular (ARMBRUST et al., 2010). O principal conceito da computação em nuvem é disponibilizar ao usuário recursos computacionais através da Internet, sob demanda e em modelo de faturamento calculado a partir da quantidade de recursos utilizados e pelo tempo de utilização (DILLON et al., 2010; MARSTON et al., 2011).

Para Rittinghouse e Ransome (2016), o uso de computação em nuvem é uma evolução natural do uso de sistemas computacionais, como exemplo, na revolução industrial, em que as fábricas deixaram de possuir sistemas próprios de geração de energia (geralmente, caldeiras e motores a vapor) e passaram e utilizar a infraestrutura pública de distribuição de energia (sobretudo elétrica e gás); as empresas tem deixado de manter seus *data centers* próprios e migrado para a nuvem.

Por outro lado, como *data centers* são desenvolvidos para suportar picos calculados, na maior parte do tempo permanecem em subutilização, assim, por meio de avanços tecnológicos, tais como virtualização de *hardware*, computação distribuída e tecnologias de Internet; é possível desenvolver tecnologias e aproveitar esse potencial subutilizado (BUYYA et al., 2010).

Computação em nuvem vem ganhando relevância na indústria e na academia (VAQUERO et al., 2008), desde grandes empresas a *startups* veem na computação em nuvem uma oportunidade de negócio, de expansão, criação de novos produtos e serviços sem um custo inicial alto, mantendo a competitividade em busca dos objetivos (MARSTON et al., 2011).

A computação em nuvem é disponibilizada através de provedores, dentre os

diversos provedores existentes atualmente no mercado, a AWS é o principal provedor de computação em nuvem, de acordo com *Synergy Research Group (2021b)*; a AWS, Amazon Web Services, possui 33% do *market share*, alcançando sozinha um terço do mercado de computação em nuvem. Lançada em 2006, a AWS de propriedade da Amazon.com oferece serviços de computação em nuvem distribuídos por várias áreas geográficas no mundo, em todos os continentes. A AWS Brasil, possui uma região de disponibilidade em São Paulo e oferece serviços de Infraestrutura (IaaS), Plataforma (PaaS) e *Software* (SaaS) (SERVICES, 2021f).

A preocupação com segurança é cada vez mais presente, uma vez que em nuvem, os dados e aplicações estão disponíveis publicamente (SHARMA et al., 2016). Um dos serviços oferecidos por provedores de computação em nuvem é o modelo de *Security-as-a-service* (Segurança como serviço), que representa a segurança oferecida como um serviço em nuvem. *Identity and Access Management* (IAM) ou gerenciamento de identidade e acesso, consiste em gerenciamento de acesso aos recursos de nuvem, concedendo acesso aos serviços com base em políticas de acesso associadas a um usuário ou grupo de usuários. Na Amazon AWS o serviço de IAM não incorre cobrança.

Code Smell são indicativos de problemas no código, que dificultam a manutenção (SJOBERG et al., 2013). Um *code smell*, de acordo com Fowler et al. (1999), em tradução livre do autor, é uma "indicação superficial que geralmente corresponde a um problema mais profundo no sistema."[1] Um *smell*, pois é fácil de sentir e farejar, como por exemplo um método muito longo, com mais de 15 linhas e várias atribuições no código, logo parece um problema. Parece, mas nem sempre é, já que alguns métodos longos são necessários e funcionam bem (YAMASHITA; MOONEN, 2013).

Os *security smells* (RAHMAN et al., 2019) são padrões de codificação que indicam um problema de segurança. Um *security smell* nem sempre leva a um problema de segurança, mas merece atenção e refatoração. A existência e persistência desses *smells* no código da aplicação aumenta a possibilidade de outro programador usar esses *scripts*, potencialmente propagando o uso de práticas de

[1] A code smell is a surface indication that usually corresponds to a deeper problem in the system.

codificação inseguras e abrindo brechas de segurança.

Uma *security smell* pode se tornar um problema de segurança, embora nem sempre seja (RAHMAN et al., 2019). Por exemplo, no AWS S3, supondo que a permissão necessária para o bom funcionamento do aplicativo necessitasse apenas de uma permissão de leitura, ao conceder uma permissão de acesso completa a um serviço de armazenamento de arquivos faria com que todos os arquivos ficassem expostos, inclusive para serem apagados. Uma abordagem viável e comumente utilizada para detecção de *smells* é a análise estática de código fonte (SABIR et al., 2019), incluindo análise de padrões linguísticos para classes, métodos e variáveis.

1.1 MOTIVAÇÃO

Com o surgimento da computação em nuvem houve uma alteração da percepção sobre arquiteturas de infraestrutura, entrega de *software* e modelos de desenvolvimento. Essa evolução, com a mudança do uso de computadores *mainframe* para o modelo cliente/servidor, a computação em nuvem engloba elementos de computação em *GRID* e autônoma em uma arquitetura inovadora (ZISSIS; LEKKAS, 2012). Os benefícios da computação em nuvem, sobretudo a implantação de aplicativos escaláveis sem investimentos em infraestrutura de *hardware*, são bastante atraentes, ao ponto que cuidados com segurança são igualmente desafiadores (CHEN et al., 2010). O problema da segurança torna-se complexo à medida que novas dimensões entram no escopo do problema, tais como: arquitetura, multi-locação, escalabilidade e dependência de serviços (ALMORSY et al., 2016).

Silva et al. (2013) apresentam um mapeamento sistemático que identifica as sete principais ameaças à segurança de computação em nuvem, sendo: (1) Abuso e uso malicioso de computação em nuvem: utilização de serviços em nuvem para ataques DDoS, que busca negação de serviço por enviar múltiplas requisições distribuídas a um servidor. (2) Interfaces e APIs inseguras: Interfaces públicas ou APIs com problemas de segurança, que permitem acesso indevido. (3) Ameaça interna: quando alguém com acesso ao provedor de computação em nuvem promove um ataque ou destruição de dados. Esta ameaça é incomum e

difícil de mitigar. (4) Problemas de tecnologia compartilhada: Ameaça que surge ao utilizar múltiplas ferramentas, inclusive de terceiros, para gerenciar a nuvem. (5) Perda ou vazamento de dados: quando uma exclusão, alteração ou apropriação de dados acontece, bem como vazamento, exposição pública, de dados sensíveis. (6) Roubo de conta ou serviço: sequestro de contas administrativas, quando os devidos cuidados, como autenticação em 2 fatores, não são tomados. (7) Perfil de risco desconhecido: é a ameaça mais explorada na literatura, já que abstrações da arquitetura e responsabilidade de manutenção dependem dos provedores.

Enquanto na academia, os problemas relacionados a segurança são pesquisados e novas propostas surgem, na indústria essas iniciativas geram esforços para o aumento da segurança; empresas como a Akamai ajudam seus clientes a mitigar problemas relacionados a nuvem, aplicações e APIs (AKAMAI, 2021). Em 2021, de acordo com a Gartner, os gastos com segurança em tecnologia da informação devem ultrapassar os 150 bilhões de dólares, sobretudo motivado pela adoção massiva da nuvem; os gastos com segurança em nuvem cresceram 41,2% e com IAM 15,6% de 2020 para 2021 (GARTNER, 2021). Neste trabalho, a AWS foi escolhida pois possui maior *marketshare* (Synergy Research Group, 2021b).

Na computação em nuvem, o excesso de privilégios em políticas de IAM aumenta o risco de segurança em caso de credenciais comprometidas (vazamento de dados, seja por problemas de segurança da aplicação ou por descuido da equipe ao publicar credenciais IAM), ameaças e uso acidental indevido. Um exemplo é o uso indevido de computação em nuvem para mineração de cripto moedas, que se faz uso de um robô para verificar credenciais expostas e com excesso de privilégios para utilizá-las e lançar instâncias de computação voltadas a mineração (RODRIGUES, 2020). Por outro lado, falta de privilégios causam problemas nas aplicações, já que as mesmas deixam de possuir acesso necessário aos serviços e recursos de nuvem (SANDERS; YUE, 2019). Como exemplo, ao tentar enviar um arquivo sem a permissão, a aplicação não conseguirá concluir a tarefa se não houver permissão, gerando exceções ou, se tratadas, mal funcionamento por não realizar a tarefa solicitada.

A partir de buscas realizadas na Internet, em buscadores e ferramentas

comumente utilizadas por desenvolvedores, como o Google e Stack Overflow, fórum comumente citado em estudos que consideram fóruns de discussão (SILVA et al., 2020); e profissionais responsáveis por implantação de aplicações em serviços de nuvem, foram identificados vários indícios de *security smells*, levando a resultados preliminares que apontam para o problema. Em busca sistemática na literatura, identificou-se trabalhos relacionados (SANDERS; YUE, 2019) (IYER; MASOUMZADEH, 2018), mas sem a devida atenção a detecção de *security smells*, com abordagens superficiais ou alternativas, que visam a definição e proposição de modelos.

1.2 OBJETIVOS

Esta seção apresenta o objetivo geral do trabalho e elenca os objetivos específicos alcançados com o desenvolvimento da pesquisa.

1.2.1 OBJETIVO GERAL

O objetivo geral deste trabalho foi desenvolver uma abordagem apoiada por uma ferramenta de análise estática de código fonte de aplicações *web*, escritas em NodeJS, PHP e Java, a fim de mapear permissões necessárias para a correta execução da aplicação e comparar com permissões concedidas ao usuário do IAM programático, para que tenha acesso aos recursos e serviços *cloud*; fornecendo um relatório de excesso e ausência de privilégios.

Este trabalho propõe uma abordagem de verificação de segurança, a ser realizada antes do *deploy* (implantação do aplicativo) com o apoio de uma ferramenta que analisa e relata a ocorrência de *security smells*, realizando análise estática do código fonte de aplicações desenvolvidas para a nuvem com uso de SDK-AWS (kit de desenvolvimento de software disponibilizados pela Amazon para a maioria das linguagens e *frameworks* disponíveis). Esta abordagem ajuda a identificar problemas de segurança ao comparar com uma biblioteca própria de problemas e as permissões concedidas ao usuário do ambiente de produção.

A ferramenta desenvolvida[2] foi avaliada e disponibilizada sob licença GPL (*General Public License*, software livre) (GNU, 2021) via repositório de código do autor para que possa ser utilizada, modificada e expandida a partir da colaboração da comunidade após a publicação deste trabalho, contribuindo com a comunidade de desenvolvimento de *software*.

1.2.2 OBJETIVOS ESPECÍFICOS

- Determinar, a partir de mineração de dados junto ao *Stack Overflow*, a incidência de respostas que induzem o desenvolvedor a conceder excesso de privilégios para o usuário IAM programático em produção, por exemplo, permitir exclusão de objetos em um *bucket* S3 em uma aplicação que possui a necessidade de apenas leitura dos objetos armazenados.
- Analisar, a partir dos dados minerados a relevância das respostas, a partir da avaliação dos usuários do fórum, e as possíveis brechas de segurança geradas.
- Definir um processo de identificação de funções a partir do SDK da AWS para as principais linguagens de programação, para a confecção da biblioteca de funções.
- Elaborar uma abordagem apoiada por ferramenta de análise estática baseada na busca de *keywords* que objetiva algumas linguagens escolhidas e que liste os privilégios necessários para a correta execução da aplicação, comparando as permissões que o usuário possui com as funções identificadas no código fonte, a partir da análise da aplicação e comparação com a biblioteca.
- Elaborar um *script* para listar as permissões concedidas ao usuário programático da aplicação em produção e verificar junto ao resultado do *script* de análise estática a ocorrência de excesso ou ausência de privilégios.
- Analisar os impactos de excesso de privilégios em aplicações *web* e os potenciais riscos.

[2]Repositório com o código fonte, ferramenta compilada com dependências e dados experimentais `https://github.com/Deizepe/CloudSSF`

1.3 ORGANIZAÇÃO DO TEXTO

Esta dissertação está organizada seguinte forma: O Capítulo 2 contém a revisão bibliográfica, com a definição de computação em nuvem; os cinco principais provedores de computação em nuvem em 2021; os principais serviços oferecidos pela Amazon AWS, tais como: S3, SES, SQS, SNS e IAM; e exemplos de aplicação em nuvem; conceitos de *Smell*; e apresentação de trabalhos relacionados. O Capítulo 3 apresenta a abordagem proposta, com a descrição do problema e as etapas do projeto: identificação do problema na prática; modelagem das APIs relacionas a políticas de segurança; desenvolvimento da ferramenta; avaliação da ferramenta. O Capítulo 4 apresenta e descreve a ferramenta produzida, sua estrutura e características. O Capítulo 5 trata da avaliação experimental, a execução do experimento, resultados e da discussão dos resultados. Por fim, o Capítulo 6 conclui o trabalho, elenca as limitações e sugestões para trabalhos futuros e a divulgação dos resultados.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentadas a fundamentação teórica e a contextualização de computação em nuvem. Na Seção 2.1, são apresentados conceitos de computação em nuvem, principais provedores, os principais serviços de computação em nuvem disponíveis na AWS e os casos de uso de serviços de computação em nuvem. Na Seção 2.2, a definição de *code smell*. Na Seção 2.3, são apresentados trabalhos relacionados, que trazem abordagens aproximadas e contribuições ao contexto da pesquisa.

2.1 COMPUTAÇÃO EM NUVEM

A computação em nuvem não é um conceito novo, de acordo com Mell et al. (2011) é uma espécie de sistema distribuído que permite acesso sob demanda a um grupo de recursos, tais como armazenamento, rede, computação e serviços, que podem ser reservados e liberados rapidamente, manual ou automaticamente, conforme a necessidade. Possui 3 principais modelos de serviços:

***Software as a Service* (SaaS)**: Prover aos usuários aplicações rodando na infraestrutura de nuvem. Aplicações disponíveis por diversas interfaces, tais como navegadores *web*, ou aplicativos. O usuário final não gerencia nem controla a infraestrutura de rede, computação, sistemas operacionais ou armazenamento, nem a aplicação (MELL et al., 2011). Por exemplo, o Google Apps ou o Microsoft Office 365, que são softwares ofertados como serviço na nuvem.

***Platform as a Service* (PaaS)**: Prover aos usuários a capacidade de disponibilizar aplicações. O usuário não controla a infraestrutura de nuvem, mas tem controle sobre a aplicação, e configuração de ambiente de hospedagem (MELL et al.,

2011). Como exemplo, o AWS Lambda.

***Infrastructure as a Service* (IaaS)**: Prover aos usuários processamento, armazenamento, rede e outros recursos de computação, a fim de implementar e executar *software* ou sistemas operacionais e aplicações. O usuário não controla a infraestrutura, mas tem controle sobre sistema operacional e algum controle sobre a rede (MELL et al., 2011). Por exemplo, o AWS EC2.

2.1.1 PRINCIPAIS PROVEDORES

Os modelos de SaaS, PaaS e IaaS são oferecidos por todos os principais provedores de computação em nuvem que, de acordo com Synergy Research Group (2021a), em ordem de fatia de mercado são: Amazon, Microsoft, Google, Alibaba e IBM, respectivamente com os serviços Amazon AWS (33% do mercado), Microsoft Azure (20%), Google GCP (8%), Alibaba Cloud (6%) e IBM Cloud (5%); os outros 28% com outros provedores. Esses cinco principais provedores são descritos a seguir.

AWS: A AWS surge em 2006 com a oferta pública de serviços de computação em nuvem da Amazon.com, empresa de varejo, que disponibiliza sua infraestrutura tecnológica para empresas e desenvolvedores. A Amazon possui infraestrutura global, com *data centers* distribuídos em todos os continentes. Por ser o foco deste trabalho, AWS é descrito em detalhes na Seção 2.1.2.

Microsoft Azure: Microsoft Azure foi apresentada ao público em outubro de 2008, na conferência de desenvolvedores da Microsoft, como uma resposta ao serviço da Amazon por parte da Microsoft. Ele oferece serviços de aplicação, computação, inteligência de negócios, análise de dados, aprendizagem de máquina, entre outros; e também está disponível globalmente (MICROSOFT, 2021).

Google GCP: O Google GCP, disponível desde 2008, disponibiliza para empresas e desenvolvedores a mesma infraestrutura utilizada pelo Google em seus serviços, como o buscador e Youtube; oferece infraestrutura global e serviços de armazenamento, aprendizado de máquina, computação, e-mails, análise de dados, banco de dados, *Big data*, IoT, entre outros (GOOGLE, 2021).

Alibaba Cloud: Lançada em 2009, a Alibaba Cloud é uma empresa do grupo

Alibaba e, embora no cenário mundial tenha apenas 6% do *marketshare* (Synergy Research Group, 2021a), é lider no segmento na China. Além de oferecer serviços tradicionais, possui serviços voltados a disponibilização de GPUs e CDN global, com ampla oferta de servidores na China, característica menos presente nos outros provedores (ALIBABA, 2021).

IBM Cloud: Surge a partir da compra da SoftLayer, empresa do segmento de IaaS, pela IBM e oferece serviços de nuvem convencionais, bem como acesso aos serviços Watson, inteligência artificial desenvolvida pela IBM (IBM, 2021).

2.1.2 SERVIÇOS DE COMPUTAÇÃO EM NUVEM DA AWS

Esta Seção elenca os principais serviços em nuvem utilizados na análise do código pela ferramenta. Estes serviços são oferecidos pela AWS Amazon Web Services. O S3 é um serviço de armazenamento de objetos. O SES um serviço de envio de e-mails. O SQS um serviço de filas gerenciadas. O SNS é um serviço de notificações, IAM é um serviço gerenciado de identidade e acesso para os serviços AWS.

S3: Amazon S3 (SERVICES, 2021d) é um serviço de armazenamento de objetos em nuvem; objetos podem ser arquivos de texto, imagem, vídeos, enfim, qualquer arquivo digital a ser armazenado. S3 é o ARN (*Amazon Resource Name*, Nome de Recurso Amazon, em tradução livre) que se dá para o *Simple Storage Service*, ou Serviço de Armazenamento Simples, em tradução livre. Os dados são armazenados como Objetos em *buckets* (baldes), que são unidades lógicas de armazenamento, tais quais os diretórios em um sistema de arquivos. Em um *bucket* pode-se armazenar qualquer quantidade de dados, e manipular esses dados, como escrever, ler e excluir os objetos. Um único objeto armazenado em um *bucket* do S3 pode ter até 5 *terabytes* de tamanho.

Os *data centers* que fornecem o serviço estão distribuídos geograficamente, como por exemplo nos Estados Unidos, São Paulo, Frankfurt, Singapura, Japão, Sydney, entre outras cidades estrategicamente selecionadas em todos os continentes. Desse modo, fornecem redundância, prevenção contra desastres naturais e *backup*

dos arquivos de modo transparente para o usuário.

O serviço de armazenamento Amazon S3 possui várias formas de restrição de acesso (SAEED et al., 2019), via IAM, *Identity and Access Management* onde é possível controlar com segurança o acesso aos recursos e serviços da AWS.

O Amazon S3 possui algumas categorias de armazenamento: (1) S3 *Standard* é o nível de serviço básico, que atende a maioria das aplicações, serviços de entrega de conteúdo, aplicativos móveis, sites; é o mais utilizado e recomendado para dados acessados frequentemente. (2) S3 *Standard-Infrequent Access*, permite armazenamento similar ao S3 Standard, porém com tempo de recuperação e disponibilidade reduzidos, disponível apenas em uma zona de disponibilidade (conjunto geográfico de *data centers*) e custo inferior. (3) S3 *Glacier* é o nível de armazenamento para arquivos duráveis e acesso infrequente, recomendado para *backup* e pode levar de minutos a horas para que o arquivo seja disponibilizado para recuperação; o custo é inferior se comparados ao S3 *Standard* e S3 *Standard-Infrequent Access*. Por fim, (4) S3 no *Outpost* é uma opção de armazenamento no local, na empresa contratante, que fornece acesso a API do S3, mantém os dados próximos aos aplicativos que os consomem.

O Amazon S3 também possibilita o versionamento de objetos, mantendo versões anteriores do mesmo objeto a fim de recuperar o mesmo arquivo em versões diferentes enviadas ao S3 ao logo do tempo, possibilitando controle de versão manual, de modo a acessar diferentes estados do arquivo ao longo do tempo e com as alterações sofridas.

SES: Amazon *Simple Email Service* (SES) é um serviço de envio de e-mails (SERVICES, 2021a) oferecido pela AWS. Com o SES é possível enviar e-mails a partir de qualquer aplicação, seja e-mail transacional, utilizado para informar clientes ou usuários de algo de seu interesse; marketing, para envio de campanhas personalizadas; ou e-mail em massa, com um envio de um mesmo conteúdo para uma grande quantidade de usuários. Permite enviar e-mails, de forma segura e escalável, a depender das necessidades da aplicação.

SQS: O AWS SQS *Simple Queue Service* (SERVICES, 2021c) é um serviço

de filas gerenciado que oferece a possibilidade de comunicação entre diferentes serviços ou microsserviços de *software*, aplicações *serverless* (sem servidor, como por exemplo AWS Lambda) ou sistemas distribuídos. Com ele é possível trocar dados entre aplicações, promovendo desacoplamento e interoperabilidade, independente da arquitetura ou linguagem utilizadas em cada diferente serviço. Por ser escalável, o SQS permite enviar, armazenar e receber mensagens entre os serviços de *software* em qualquer quantidade, sem o risco de perda de mensagens, e com alta disponibilidade.

O SQS disponibiliza dois tipos de filas de mensagens, (1) Filas padrão, que possibilitam alto volume de entrada e saída de mensagens, com um número quase ilimitado de transações por segundo, com ao menos uma entrega e, quase sempre, na ordem em que foram inseridas e (2) filas FIFO, por padrão, suportam até 300 transações por segundo de mensagem única ou 3000, se processadas em lotes de 10, limites que podem ser aumentados, que garante que as mensagens sejam processadas uma única vez na mesma ordem que foram adicionadas à fila.

As mensagens podem receber até 256 KB de de texto em qualquer formato (e.g. JSON, XML). Um exemplo de *software* não gerenciado que oferece, entre outras, funcionalidades similares ao Amazon SQS é o Apache Kafka (APACHE, 2021).

SNS: O AWS SNS *Simple Notification Service* (SERVICES, 2021b) é um serviço de mensagens gerenciado que oferece comunicação em duas vias, Aplicação para Aplicação (A2A) e Aplicação para Pessoa (A2P). Em (1) A2A o SNS oferece funcionalidade no padrão *Publish/Subscribe* com base em *push* (notificações enviadas para aplicações) e N para N, muitos para muitos, entre sistemas distribuídos, *serverless* orientado a eventos e microsserviços. Com SNS é possível enviar mensagens para sistemas assinantes, entre eles filas do SQS, funções sem servidor do AWS Lambda e HTTPS. Em A2P é possível enviar mensagens para os usuários utilizando SMS, e-mail ou notificações *push* em dispositivos móveis, em larga escala.

IAM: O AWS IAM *Identity and Access Management* é um serviço de gerenciamento de identidade e acesso a AWS (SERVICES, 2021e). Ele possibilita o gerenciamento de acesso a serviços e recursos da AWS. Nele é possível criar e gerenciar grupos e usuários e, utilizando permissões, negar ou conceder acesso a

recursos AWS, como por exemplo, todos os serviços descritos anteriormente.

De acordo com Sharma et al. (2016), um modelo de segurança baseado em IAM como serviço, concentra-se em: autenticação, autorização, administração de identidades e auditoria, por meio de logs. Sua principal atribuição é verificar a identidade e conceder o acesso correto aos serviços e recursos protegidos no ambiente de nuvem.

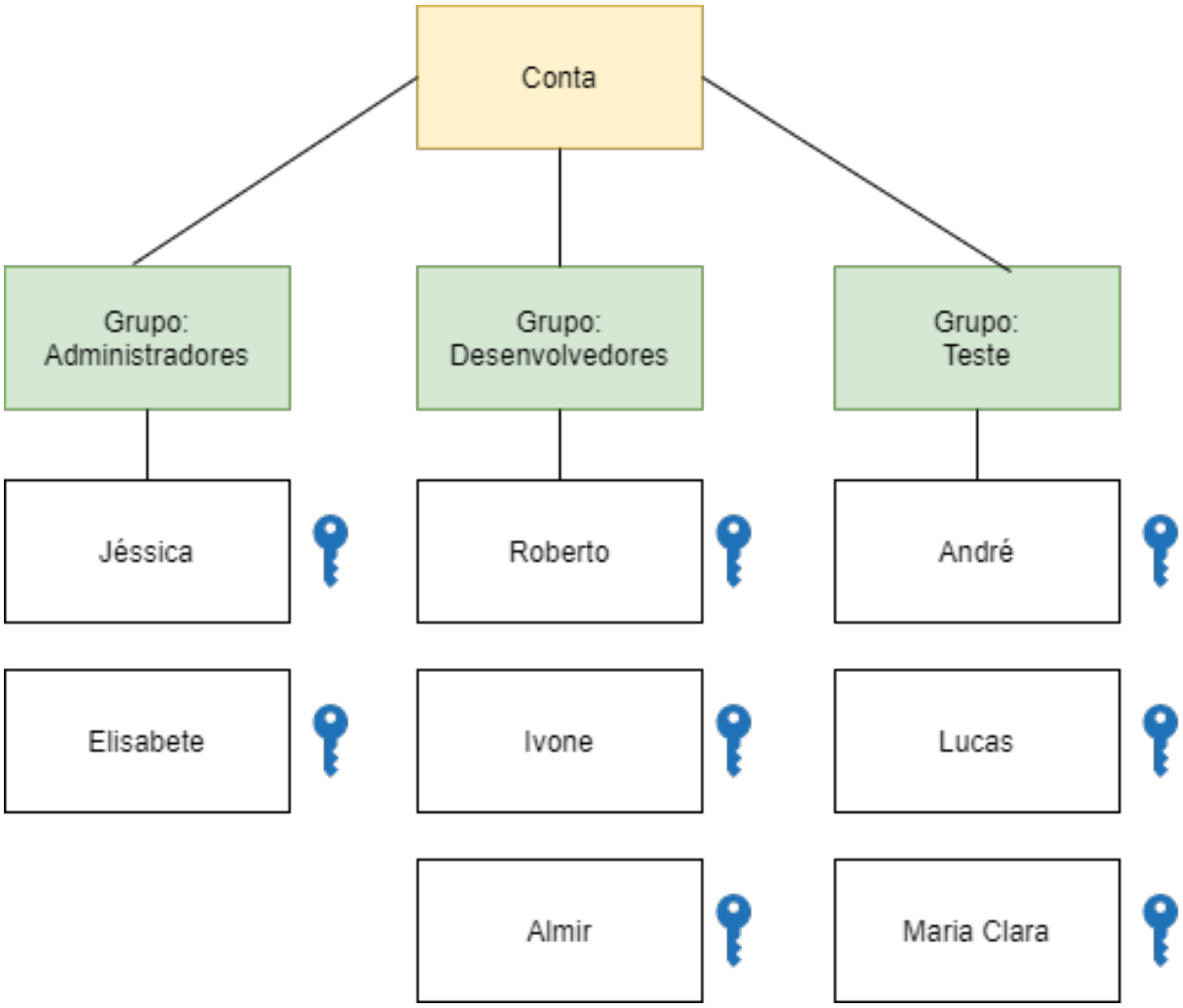
A AWS disponibiliza um catálogo de políticas que abrange genericamente todos os serviços em nuvem que oferece. Ainda é possível criar políticas personalizadas, a fim de atender especificidades do projeto de *software* a ser implantado (SERVICES, 2021e).

No modelo do IAM, um grupo é um espaço que reúne usuários com níveis de acesso similares, possibilitado a atribuição de políticas a grupos, não a usuários individualmente, o que facilita a gestão, pois ao incluir um novo usuário em um grupo, este já herda todas as políticas associadas ao mesmo (SERVICES, 2021e).

Políticas são criadas pelo administrador e vinculadas a identidades, que podem ser usuários, grupos de usuários ou funções; e definem as permissões para uma ação (SERVICES, 2021e). A políticas são descritas no formato JSON e definem o que a identidade pode executar e a quais serviços possui acesso.

Na Figura 1 vê-se um exemplo de IAM em uma conta AWS, com a conta possuindo a chave principal, 3 grupos, Administradores, Desenvolvedores e Teste com seus respectivos usuários, chaves e permissões de acesso. As políticas, que definem as permissões de acesso, são associadas aos grupos.

Figura 1: Grupos e Usuários AWS IAM. Adaptado de (SERVICES, 2021e).



A Figura 2 ilustra a política “arn:aws:iam::aws:policy/AmazonS3FullAccess”, em formato JSON, que faz parte do catálogo de políticas (SERVICES, 2021e) disponibilizado pela AWS, que tem como função armazenar modelos de políticas, na sua versão “2012-10-1” (é possível manter até 5 versões de uma política) e concede acesso “*Allow*” ao ARN (recurso) S3 (Armazenamento) e S3 Lambda (armazenamento a partir de funções *serverless*), em todos os *buckets* “*Resource*”: “*”.

Figura 2: Política Amazon AWS: S3FullAccess. Adaptado de (SERVICES, 2021e).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:*", "s3-object-lambda:*"],
      "Resource": "*"
    }
  ]
}
```

Um exemplo de criação de uma política personalizada, considerando uma conta com diversos *buckets*, que são espaços de armazenamento de arquivos, é substituir a linha “Resource”: “*”, por “Resource”: “arn:aws:s3:::bucketA/*”, isso concede acesso apenas a um determinado *bucket* (bucketA) aos grupos e usuários aos quais esta política personalizada está associada, configuradas via painel do usuário (SERVICES, 2021e).

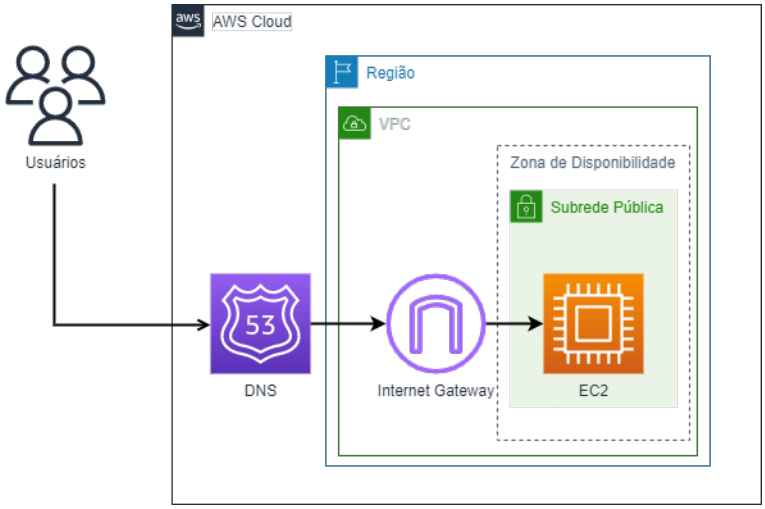
2.1.3 CASO DE USO DE SERVIÇOS DE COMPUTAÇÃO EM NUVEM

Esta subseção tem por objetivo exemplificar e demonstrar a interação entre serviços ofertados pela AWS com a finalidade de disponibilizar publicamente na internet um site utilizando o CMS WordPress, que é um popular gerenciador de conteúdo. Ao utilizar serviços de computação em nuvem, o desenvolvedor ou responsável pela implantação de serviços em nuvem, arquiteto, pode utilizar um modelo de serviços que atenda as necessidades da aplicação e do modelo de negócio.

A Figura 3 ilustra um exemplo simples da utilização da AWS Cloud para hospedagem de um site, sem um bom aproveitamento de uma arquitetura *cloud*. No exemplo, que utiliza o serviço de DNS Route53 como gerenciador de DNS, apontando para um endereço IP público, de uma máquina virtual do serviço EC2, que por sua vez suporta o servidor *web* (Apache), de aplicação (PHP), banco de dados relacional (MySQL) e no sistema de arquivo são armazenados os arquivos estáticos (Fotos, Documentos disponibilizados para download em PDF). Tal modelo assemelha-se ao

modelo de VPS, Servidor Virtual Privado, amplamente comercializado por empresas de hospedagem, disponibilizando todos os recursos em uma única máquina virtual.

Figura 3: Hospedagem de um site Wordpress usando uma VM.

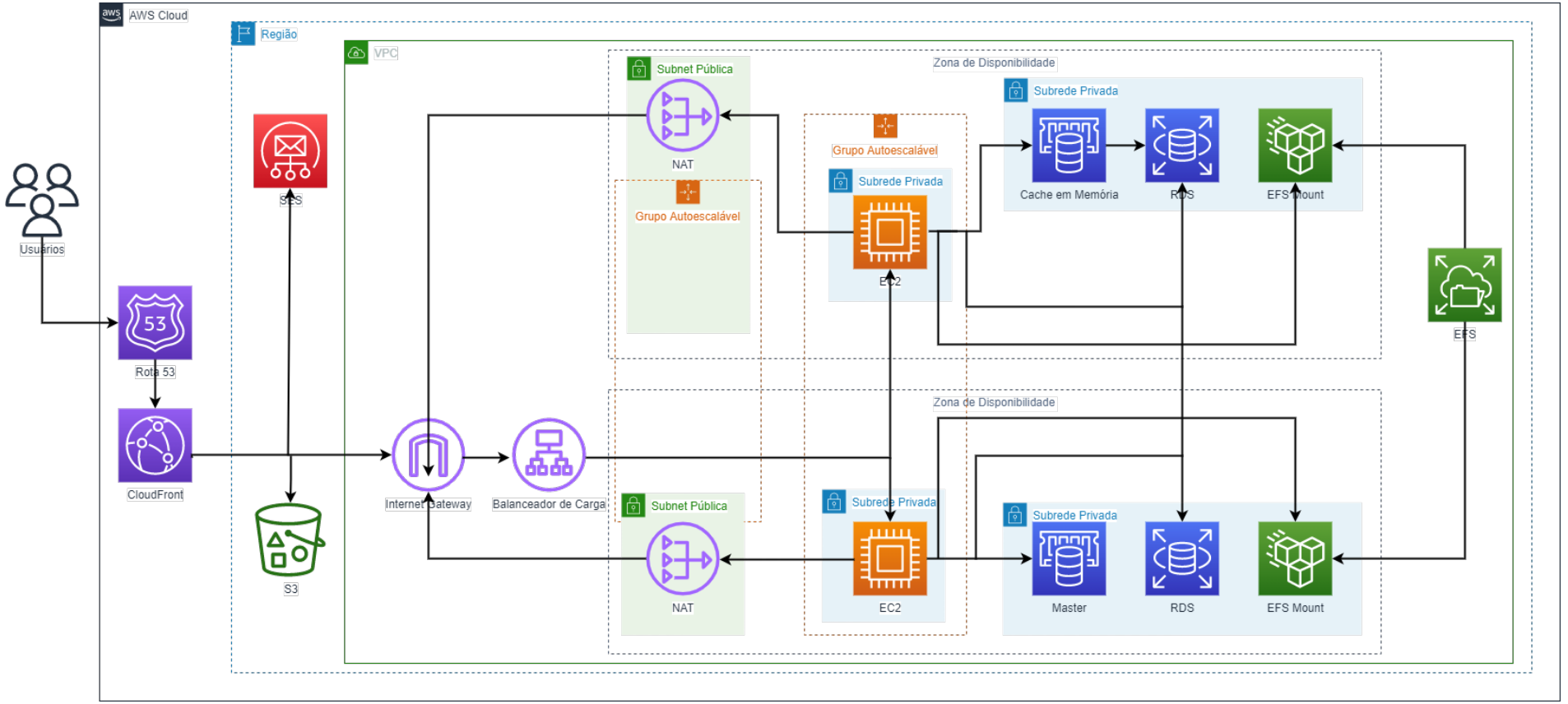


Fonte: O Autor

Este exemplo tem como vantagem o baixo custo, porém são muitas as desvantagens por não utilizar recursos e serviços nativos de Nuvem: não escala automaticamente, ou seja, ao receber muitas requisições o site ficará indisponível; A manutenção é manual, dependendo de acesso ao servidor por SSH para manutenção, atualização de segurança e alteração de recursos; segurança, já que todos os recursos estarão em uma máquina, uma vulnerabilidade exporia toda a aplicação.

A mesma aplicação de exemplo (um site Wordpress) pode ser hospedada em uma arquitetura que aproveita os recursos de nuvem, conforme ilustra a Figura 4. O site utiliza o serviço Route 53 como gerenciador de DNS. Toda a aplicação fica sob a CloudFront, para melhorar a segurança e fornecer *cache*. Utiliza *bucket* S3 (com permissões de apenas leitura) para armazenar arquivos estáticos, como fotos, documentos disponibilizados para *download* em PDF, o que possibilita distribuição em CDN global, SES para envio de e-mails (Recuperação de senhas, *Newsletter*). Com um balanceador de carga dividindo a carga de trabalho, inicialmente, para dois servidores de aplicação que cuidam apenas deste escopo, atrás de um serviço de NAT, auto escalável, podendo ligar e desligar máquinas EC2 conforme o fluxo de acesso do site, para que responda todas as requisições.

Figura 4: Hospedagem de um site Wordpress usando vários serviços de nuvem.



Fonte: O Autor

O banco de dados, gerenciado pela AWS, no serviço de banco de dados relacional RDS, possui *cache* em memória e possui cópias de redundância, uma *master* e outras somente leitura, para replicar e ampliar a capacidade de leitura, para entrega de conteúdo para a aplicação. Também o sistema de arquivos do sistema operacional está em um sistema de arquivos elástico da AWS, o EFS, a fim de montar conforme a quantidade de máquinas ligadas e desmontar, assim que o balanceador de carga reduzir a quantidade de instâncias.

Tanto os servidores de aplicação, banco em memória e banco de dados relacional estão em uma sub rede privada, o que não fornece acesso público e amplia a segurança dos dados e VMs.

A comparação entre os 2 exemplos, que na prática disponibilizam uma mesma aplicação na Internet (um site WordPress) demonstra as diferenças entre um projeto simples e complexo de arquitetura de computação em nuvem. Dependendo das necessidades o primeiro exemplo (Figura 3) é suficiente, como em sites com poucos acessos simultâneos, ou em fase de testes; mas uma aplicação robusta, escalável e segura, é demonstrada no segundo exemplo (Figura 4). Tanto no exemplo simples quanto complexo, ausência ou excesso de permissões causam erros e vulnerabilidades, já que, em casos como os exemplificados, as políticas IAM são configuradas por pessoas com pouca experiência.

2.2 CODE SMELL

Code smells, são indicadores de problemas de qualidade em software, que podem dificultar a leitura e manutenção do código. Os cheiros normalmente não são *bugs* nem impedem diretamente a execução do software; são estruturas que indicam violações de princípios de desenvolvimento e impactam negativamente na qualidade (SURYANARAYANA et al., 2014). Determinar o que é ou não um *smell* é uma tarefa que envolve subjetividade, variando de acordo com a linguagem, metodologia e padrões de desenvolvimento envolvidos.

O termo cunhado por Kent Beck e Martin Fowler (FOWLER, 1999), apresenta possibilidades de problemas em projetos de *software*. Um código refatorado tende

a diminuir a presença de *smells*, e uma nova refatoração é necessária caso sejam novamente identificados.

Code smells podem ser divididos como: (1) *smells* a nível de aplicação: código duplicado, variáveis reaproveitadas, nomes incondizentes; (2) *smells* a nível de classe: classe grande demais, complexidade ciclomática, variáveis órfãs; e (3) *smells* a nível de métodos: muitos parâmetros, métodos longos, muitos comentários, muitos retornos, entre outros (FOWLER, 1999).

Um *security smell* em um software pode ser a atribuição desnecessária de permissões no uso de uma API, fornecendo excesso de privilégios ao usuário autenticado utilizado no software. Este *smell* não causa diretamente uma violação de segurança, mas em caso de exposição da aplicação, amplificam-se as possibilidades de danos. Como exemplo de refatoração de *security smell*, em uma API de envio de e-mails, limitar os endereços autorizados a enviar mensagens, a quantidade de mensagens enviadas por período de tempo e impedir o cadastramento de novos endereços, pode mitigar ou impedir o uso desta API para envio de *spam* em caso de exposição não autorizada da API.

2.3 TRABALHOS RELACIONADOS

Rahman et al. (2019) define *security smells* como padrões de codificação que indicam um problema de segurança e apresenta exemplos e técnicas com objetivo de ajudar os profissionais a evitar práticas de desenvolvimento de códigos inseguros ao desenvolver a infraestrutura como *scripts* de código, por meio de um estudo empírico de *security smells* nesses *scripts*. Os autores analisaram 293 repositórios de código aberto, identificando 21201 ocorrências de *security smells*, dentre elas, por exemplo, 1326 ocorrências de senhas direto no código. Enviaram aleatoriamente 1000 relatórios de *bug* aos desenvolvedores, obtiveram 21,2% de respostas e 14,8% foram corrigidos. Com isso conclui-se que *security smells* podem ter vida longa em aplicações.

Sanders e Yue (2019) propõem um modelo ABAC, modelo de controle de acesso baseado em atributos, para a criação de políticas IAM personalizadas com base nas necessidades de atributos de usuários, operações, recursos e ambiente. Os

autores apresentam a abordagem de minerar regras a partir dos registros de auditoria, para a criação de políticas ABAC, levando em consideração privilégios mínimos e métodos de otimização, para evitar excesso de privilégios. A partir da mineração de uma grande quantidade de registros de logs (quase 5 milhões), o artigo apresenta as políticas desenvolvidas automaticamente pela ferramenta proposta. O artigo tem como principal objetivo a implantação automática de modelos ABAC e maior proteção dos sistemas e suas funcionalidades, evitando falta e excesso de privilégios.

Iyer e Masoumzadeh (2018) também partem do modelo ABAC, utilizam um algoritmo de mineração de logs para identificar e gerar políticas IAM com base em necessidades de atributos, vão além de permissões e propõem uma abordagem negativa, ou seja, ao invés de atribuir apenas políticas de permissões, o algoritmo de mineração analisa quais serviços não são utilizados e nega acesso aos mesmos. Segundo os autores, essa abordagem melhora a qualidade das políticas ABAC geradas, diminui o tempo de mineração e oferece melhor qualidade no controle de acesso, evitando excessos de privilégios. Os resultados são analisados com experimentos de autorizações positivas e com autorizações positivas e negativas.

Ao contrário do proposto por Sanders e Yue (2019) e Iyer e Masoumzadeh (2018) que propõem a geração de políticas e análise de excesso e/ou falta de privilégios a partir da análise de logs, ou seja, com a aplicação publicada, em produção e gerando *logs*, este trabalho pretende identificar a falta e/ou excesso de privilégios antes da publicação e implantação da aplicação (*deploy*), ou seja, antes de expor a aplicação a potenciais riscos.

2.4 CONSIDERAÇÕES FINAIS

Neste capítulo foram abordados conceitos de computação em nuvem, apresentando os principais provedores. Na Seção de serviços de computação em nuvem, foram apresentados os principais serviços disponibilizados pela AWS, tais como S3, SES, SQS, SNS e IAM. Ao final, foram apresentados exemplos de casos de uso de aplicação em nuvem, um simples e outro utilizando serviços nativos de nuvem para publicar a mesma aplicação. Também foi abordada a definição de *Code*

Smells, *Security Smells*, bem como trabalhos relacionados, que mineram dados via análise de *logs*, após a publicação da aplicação e não antes do *deploy*, como é o caso da abordagem proposta neste trabalho.

3 MÉTODO E ABORDAGEM PROPOSTA

No contexto de aplicações hospedadas em nuvem, permissões e políticas utilizando IAM servem para definir acesso a serviços e recursos. A AWS, provedor selecionado para este estudo, fornece IAM como serviço.

Este trabalho buscou identificar e formular uma biblioteca de casos de uso da SDK da AWS disponibilizada em diferentes linguagens de programação, que serviu de base para a ferramenta; desenvolveu uma ferramenta de análise estática, a fim de mapear permissões necessárias para a correta execução da aplicação e comparou com permissões concedidas ao usuário do IAM programático, fornecendo ao desenvolvedor um relatório de excesso e ausência de privilégios.

Neste capítulo é apresentada a abordagem proposta. Na seção 3.1 é apresentada a descrição do problema; na Seção 3.2 são apresentadas as etapas do projeto, divididas em 4, identificação do problema, modelagens das APIs e SDK, desenvolvimento da ferramenta e avaliação da abordagem proposta e ferramenta.

3.1 DESCRIÇÃO DO PROBLEMA

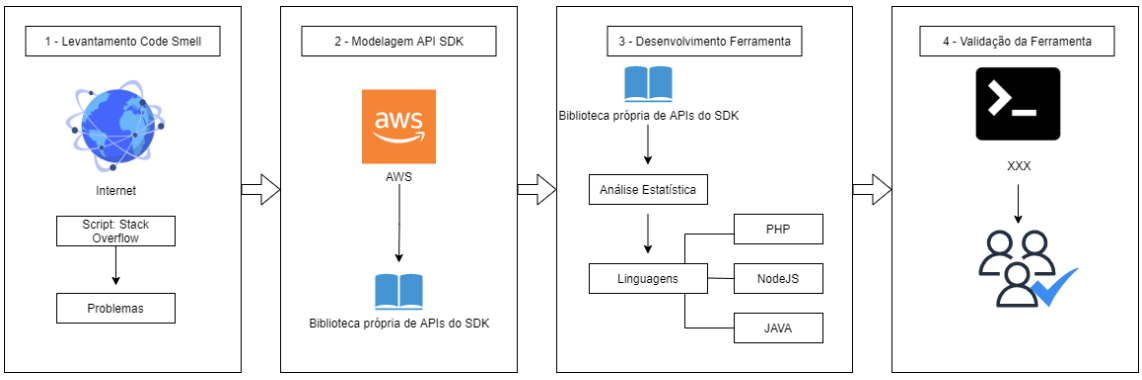
Considerando que excesso ou ausência de privilégios acarretam desde a exposição desnecessária de recursos até o mal funcionamento da aplicação, respectivamente, e que as abordagens encontradas em trabalhos relacionados (SANDERS; YUE, 2019; IYER; MASOUMZADEH, 2018) focam na análise de *logs* após a implantação do *software* em produção, e ABAC, este projeto busca resolver o problema (excesso e ausência de privilégios em usuários IAM de aplicações em produção) e apontar *security smells* ao desenvolvedor antes da disponibilização da aplicação, ou versão, em ambiente de produção, o *deploy*.

3.2 ETAPAS DO PROJETO

Este projeto está dividido em 4 partes principais: a subseção 3.2.1 apresenta a etapa que buscou identificar *Code Smells* em postagens de ajuda a desenvolvedores na plataforma SteckOverflow. A subseção 3.2.2, aborda a modelagem do SDK (Kit de desenvolvimento de *Software*) da AWS, e elaboração de uma biblioteca própria com possíveis *security smells*, a identificar em políticas de permissão da AWS. A subseção 3.2.3 consiste em, a partir da biblioteca criada na etapa anterior, apresentar a arquitetura da ferramenta de análise estática de código fonte em interface CLI, linha de comando, que identificou *Security Smells* nas aplicações antes do *deploy*, para as linguagens PHP, NodeJS e JAVA, por serem as mais utilizadas na web (GITHUB, 2021). A subseção 3.2.4 apresenta a etapa que validou a abordagem e ferramenta junto a experimentos de publicação de *softwares* livres, em nuvem AWS, em diferentes linguagens de programação e diferentes cenários, simulando excesso e ausência de privilégios nos usuários IAM da AWS, com políticas genéricas e personalizadas.

A Figura 5 ilustra o processo descrito das etapas do projeto, suas entradas e produtos. São quatro etapas, levantamento de *code smell*, modelagem da API SDK, desenvolvimento da ferramenta e avaliação da ferramenta.

Figura 5: Etapas do projeto.



Fonte: O Autor

3.2.1 IDENTIFICAÇÃO DO PROBLEMA NA PRÁTICA

Nesta etapa, foi realizada a pesquisa no site Stack Overflow, a partir de uma aplicação de mineração de dados desenvolvida em Python, em 400 *links*, dos 10 primeiros resultados para 10 *strings* de pesquisas, apresentadas na Tabela 1, em cada um dos 4 serviços (SES, S3, SQS, SQS) abordados nesse trabalho. Esta etapa foi auxiliada por um *script* de mineração produzido pelo autor, em Python, e disponibilizado em seu repositório público no GitHub, que automatiza a busca utilizando *string* de pesquisa na API do *Google Search* cruzando os resultados com palavras chaves encontradas no texto (mineração e análise do DOM da página). A fim de evitar falsos positivos ou falsos negativos, todos os resultados foram analisados manualmente e após exclusão de resultados incorretos, os dados foram agrupados na Tabela 1.

Tabela 1: Resultados do *script* de busca no Stack Overflow

Serv.	**String de busca**	**String de Security Code Smell**				
		's3:*'	'ses:*'	'sqs:*'	'sns:*'	'full access'
S3	aws s3 error upload file laravel	0	0	0	0	0
	aws s3 AccessDenied	2	0	0	0	3
	aws s3 bucket permissions	2	0	0	0	2
	aws iam s3	3	0	0	0	1
	putObject s3	0	0	0	0	1
	error send file aws s3	0	0	0	0	0
	s3 Policy has invalid action - s3:ListAllMyBuckets	0	0	0	0	0
	this request does not support credentials s3	0	0	0	0	0
	aws s3 iam credentials	0	0	0	0	0

Continua na próxima página

Tabela 1: Resultados do *script* de busca no Stack Overflow (Continuação)

	s3 CredentialsNotSupported	0	0	0	0	0
SES	aws ses error sending email	0	0	0	0	0
	aws ses AccessDenied	0	0	0	0	2
	aws ses domain	0	0	0	0	0
	aws ses permissions	0	0	0	0	1
	list email ses	0	0	0	0	0
	list domain ses	0	0	0	0	0
	ses policy has invalid action	0	0	0	1	0
	aws ses iam credentials	0	0	0	0	2
	ses CredentialsNotSupported	0	0	0	0	0
	MailFromDomainNotVerified ses	0	0	0	0	0
SQS	sqs NotAuthorized	0	0	2	0	3
	aws sqs error	0	0	0	0	0
	aws sqs AccessDenied	0	0	3	0	2
	aws sqs permissions	0	0	2	0	0
	aws sqs iam	1	0	4	0	0
	sqs error queue	0	0	1	0	0
	aws sqs InvalidAction	0	0	0	0	1
	aws sqs NotAuthorized	0	0	1	2	5
	sqs MissingParameter	0	0	0	0	0
	sqs InvalidClientTokenId	0	0	0	0	1
SNS	aws sns AccessDeniedException	1	0	0	1	2
	aws sns error	0	0	0	0	0

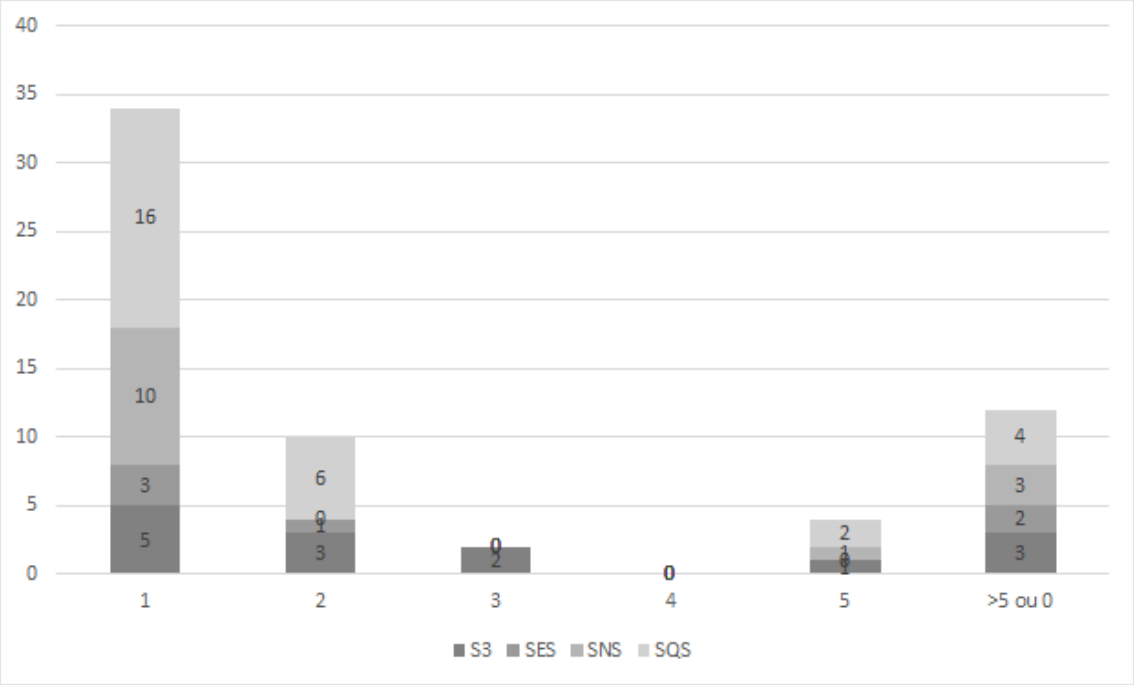
Continua na próxima página

Tabela 1: Resultados do *script* de busca no Stack Overflow (Continuação)

aws sns AccessDenied	0	0	1	0	0
aws sns permissions	1	0	1	1	1
aws sns iam	0	0	0	2	0
aws sns error notification	0	0	0	0	0
aws sns message delivery error	0	0	0	0	0
sns Publish error	0	0	0	0	0
sns topic error	1	0	0	1	1
sns InvalidAction	0	0	0	0	0

A Tabela 1 apresenta a quantidade de ocorrências de políticas que geram excesso de privilégios nas respostas melhor avaliadas ou únicas às perguntas realizadas no Stack Overflow referente a problemas de privilégios encontrados na execução de *software* em nuvem AWS. As *strings* de busca são apresentadas nas linhas, agrupadas pelos 4 serviços pesquisados, e os *security code smells* apresentados nas colunas. Os números nas intersecções contam as ocorrências de *code smells* nas respostas.

Figura 6: Ocorrência de resultados de acordo com a posição da resposta no Stack Overflow.



Fonte: O Autor

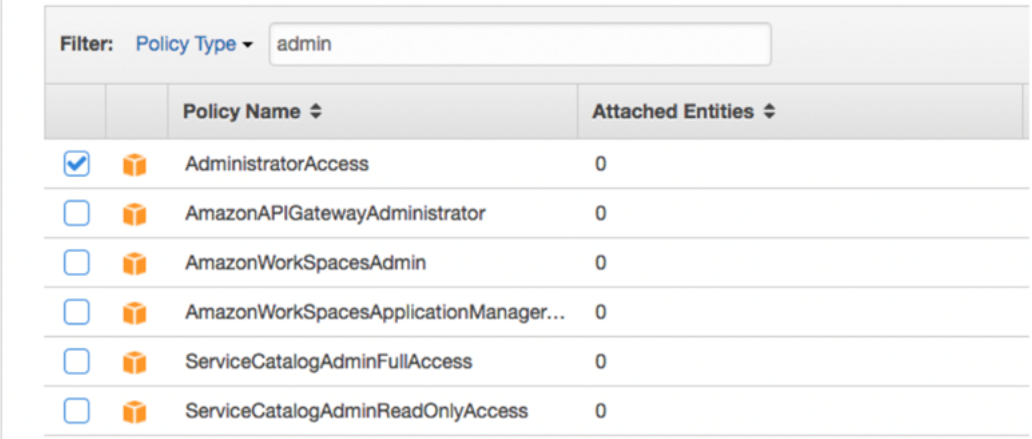
O gráfico da Figura 6 apresenta no eixo x a posição da resposta que contém o *smell* em relação às outras respostas, de acordo com a votação dos usuários da plataforma; no eixo y, a quantidade de respostas em cada posição. Observou-se a ocorrência predominante de *smell* na resposta melhor avaliada. Entre as respostas que receberam votos, 14,2% apresentavam sugestões e soluções incorretas ou que causariam *security smells*.

Com base nas ocorrências em uma busca simplificada na plataforma, foi possível concluir preliminarmente que as respostas com soluções que geram possíveis problemas de excesso de privilégio são uma constante no fórum Stack Overflow.

Considerando o *Stack Overflow* como um dos principais fóruns (ACAR et al., 2016) que os desenvolvedores utilizam com a finalidade de encontrar uma solução para os problemas encontrados durante o desenvolvimento e publicação de aplicações; e que a principal fonte de tráfego vem de buscadores, como o Google, a partir de um *script* de mineração foram localizados algumas respostas que usuários forneceram no fórum de discussão.

Figura 7: Resposta com sugestão para que o desenvolvedor conceda privilégios de Acesso Administrativo para o usuário programático usado em ambiente de produção da aplicação.

The correct answer to this is definitely: IAM -> Users -> "Attach User Policy" and make your user admin



Fonte: `https://stackoverflow.com/questions/11147961/ruby-amazon-s3-access-denied-when-listing-buckets`. Acesso em: 18 de julho de 2022.

Na Figura 7, retirada de uma das respostas encontradas no Stack Overflow pelo *script* de mineração[1], encontra-se um erro bastante usual e ao mesmo tempo grave, que possui bastante recorrência em indicação como resposta para solução de questões de desenvolvedores que procuram o site e que encontram problemas com permissões na AWS. Esta permissão concede privilégios administrativos totais sobre a conta, inclusive a possibilidade de criar novos usuários, apagar usuários existentes e alocar recursos. Em uma situação hipotética de usuário comprometido, é possível gerar sequestro da conta, utilização de recursos computacionais para mineração de cripto moedas, envio de *spam* via SES, injeção de SQL, vazamento de informações e arquivos sigilosos e diversos outros usos maliciosos.

[1]Repositório com o código do *script* em Jupyter Notebook e as perguntas executadas no Google `https://github.com/Deizepe/MestradoScriptPyStackOverflow/blob/main/GoogleBusca.ipynb`

Figura 8: Resposta com excesso de privilégios em fila SQS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:*"
            ],
            "Resource": [
                "arn:aws:sqs:us-east-1:*:*"
            ],
            "Effect": "Allow",
        }
    ]
}
```

Fonte: https://stackoverflow.com/questions/47034903/an-error-occurred-invalidclienttokenid-when-calling-the-assumerole-operation/50395014. Acesso em: 12 de julho de 2022.

Na Figura 8, também retirada de uma das respostas encontradas no Stack Overflow através do *script* de mineração, é possível encontrar uma resposta sugerindo ao desenvolvedor que realizou a pergunta que dê acesso total às filas SQS em toda a região east-1 (*North Virginia*) para o usuário da aplicação. Essa sugestão, que apesar de possuir menor gravidade em relação a da Figura 7, representa riscos de excesso de privilégios na AWS.

Na Figura 9 é possível encontrar uma peculiaridade. Esta resposta foi encontrada no Stack Overflow pelo *script* de mineração e, durante o processo de análise, foi indicada como falso positivo. A resposta aponta um problema (na primeira metade, numa resposta simplista que resolve o problema apresentado na pergunta) mas logo a seguir indica que é uma boa prática não conceder excesso de permissões, dando ao usuário apenas acesso estritamente necessário para a execução da aplicação.

No caso apresentado pelo autor da pergunta e corretamente apresentado (na segunda metade da resposta), permite apenas a listagem do conteúdo de um *bucket* e leitura de objetos armazenados nele. Nesse caso, com esse usuário, não é possível incluir ou excluir arquivos desse *bucket*.

Figura 9: Resposta que aponta excesso de privilégio e faz sugestão de correção.

Slightly modifying your policy would look like this:

```
{
  "Version": "version_id",
  "Statement": [
    {
        "Sid": "some_id",
        "Effect": "Allow",
        "Action": [
            "s3:*"
        ],
        "Resource": [
            "arn:aws:s3:::bucketname",
            "arn:aws:s3:::bucketname/*"
        ]
    }
  ]
}
```

However, that probably gives more permission than is needed. Following the AWS IAM best practice of Granting Least Privilege would look something like this:

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "s3:ListBucket"
          ],
          "Resource": [
              "arn:aws:s3:::bucketname"
          ]
      },
      {
          "Effect": "Allow",
          "Action": [
              "s3:GetObject"
          ],
          "Resource": [
              "arn:aws:s3:::bucketname/*"
          ]
      }
  ]
}
```

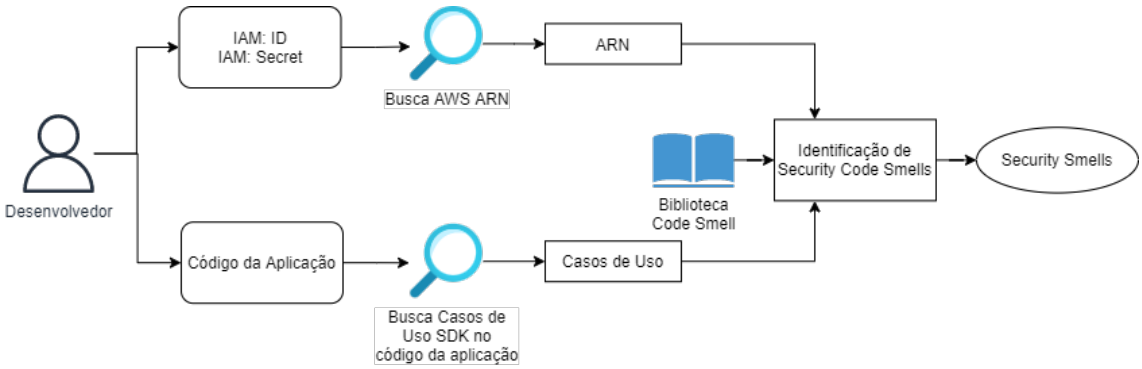
Fonte: https://stackoverflow.com/questions/38774798/accessdenied-for-listobjects-for-s3-bucket-when-permissions-are-s3. Acesso em: 12 de julho de 2022.

3.2.2 MODELAGEM DAS APIS RELACIONADAS A POLÍTICAS DE SEGURANÇA

Esta etapa mapeou os métodos do SDK da Amazon AWS nas linguagens PHP, NodeJS e JAVA (levando em consideração os diferentes padrões de nomeação, como por exemplo: sendEmail = SendEmail = send_email), relacionou com as APIs da Amazon AWS e os respectivos serviços, e construiu uma biblioteca para apoio no processo de análise estática realizado pela ferramenta.

3.2.3 ARQUITETURA DA FERRAMENTA

Figura 10: Funcionamento da Ferramenta.



Fonte: O Autor

Esta etapa refere-se a construção de uma ferramenta de análise estática de código fonte, com interface CLI, linha de comando, que foi desenvolvida em C#, que tem como finalidade identificar *Security Smells* em políticas de segurança de serviço de computação em nuvem AWS. A ferramenta é capaz de operar com projetos desenvolvidos nas linguagens PHP, NodeJS e JAVA.

Conforme ilustrado na Figura 10, o desenvolvedor deve instalar o pacote da ferramenta no ambiente de desenvolvimento, fornecer as credenciais IAM Amazon AWS ID e Secret do usuário de produção, o local onde o código da aplicação a ser analisada está e a linguagem utilizada para o desenvolvimento; a ferramenta realiza inicialmente uma busca pelas políticas e permissões concedidas àquele usuário junto a Amazon AWS e posteriormente, via mineração e análise estática no código fonte da aplicação, identifica os casos de uso da SDK e, com base na biblioteca de *code smells*

desenvolvida, cruza os dados obtidos e identifica, apresentanado ao desenvolvedor as *security smells* identificadas.

3.2.4 AVALIAÇÃO DA ABORDAGEM PROPOSTA E FERRAMENTA

A fim de validar a abordagem proposta e a ferramenta que foi desenvolvida, foi realizada a implantação de *software* livres (como exemplo: Wordpress, Moodle, LaraERP, Akaunting) em infraestrutura de nuvem, usando serviços de nuvem para gerenciar computação (EC2), armazenamento (S3), envio de e-mails (SES), notificações de falhas (SNS) e filas (SQS). Utilizando as políticas genéricas da Amazon, casos com FullAccess e simulações com políticas personalizadas, ocorrendo carência de privilégios, a fim de verificar a precisão da ferramenta em identificar excessos e ausência de privilégios necessários para a correta execução da aplicação a partir da análise estática de código, verificação de uso de métodos da API com o SDK utilizado na aplicação comparado ao catalogo de *smells* identificado previamente.

3.3 ABORDAGEM

Com base no método descrito acima, a abordagem consiste em analisar o código fonte uma aplicação pronta para ser disponibilizada em ambiente de produção, junto ao usuário programático IAM disponibilizado para o *deploy*, instanciada na ferramenta desenvolvida a fim de encontrar *smells* e apresentar ao desenvolvedor, tanto os contidos no código fonte da aplicação, quanto os contidos na nuvem, afim de que esse possa avaliar e mitigar possíveis problemas de seguranças.

3.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a abordagem proposta, a descrição do problema e as etapas do projeto: identificação do problema, modelagem das APIs e criação da biblioteca de *smells*, desenvolvimento da ferramenta e avaliação. O capítulo 4 descreve a ferramenta desenvolvida.

4 CLOUDSSF (CLOUD SECURITY SMELL FINDER)

Este capítulo apresenta a ferramenta desenvolvida para apoiar a abordagem proposta neste trabalho, como utilizá-la, as tecnologias utilizadas e descreve a biblioteca de *Code Smells* e sua estrutura.

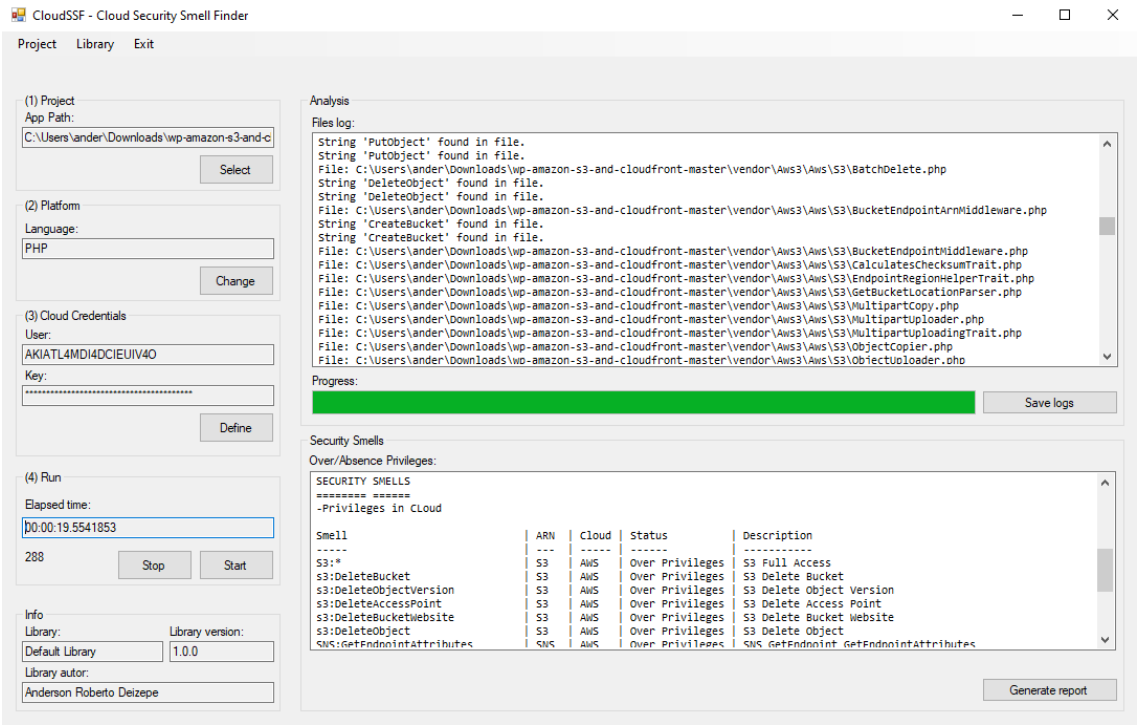
4.1 IMPLEMENTAÇÃO DA FERRAMENTA

Para a execução deste projeto foi elaborada uma ferramenta chamada CloudSSF - *Cloud Security Smell Finder* que implementa a abordagem proposta para identificação de *Security Smells* em aplicações web em fase de produção. Esta ferramenta permite, a partir do fornecimento de dados de acesso do usuário IAM que será utilizado pela aplicação em ambiente de produção, identificar ausência e excesso de privilégios a partir da utilização da biblioteca de *code smells*.

Para a implementação da ferramenta foram utilizadas as seguintes tecnologias: Linguagem de programação C#, *framework* .Net6.0, banco de dados em CSV e JSON, SQLite e ORM EntityFramework. A IDE utilizada para desenvolvimento foi o Microsoft Visual Studio 2022.

A ferramenta CloudSSF verifica a partir das credenciais do ambiente de produção, quais permissões o usuário IAM possui e, cruzando com os métodos da biblioteca de *Code Smells*, fornece um relatório detalhado de possíveis *Security Smells* detectados, tanto em privilégios em excesso, quanto ausentes no projeto submetido ao teste.

Figura 11: Ferramenta CloudSSF.



Fonte: O Autor

Na tela principal ilustrada na Figura 11, o usuário deve configurar a ferramenta para iniciar o teste em um projeto ou nova versão de um projeto prestes a ser disponibilizado em ambiente de produção. Na seção 1, *Project*, o usuário deve selecionar a pasta no sistema de arquivos que contém o projeto. Na seção 2, *Platform*, deve ser escolhida a linguagem de programação principal do projeto a ser testado. Na seção 3, *Cloud Credentials*, o usuário deve informar as credenciais que serão utilizadas pela aplicação em ambiente de produção. Por fim, na seção 4, *Run*, o usuário deve clicar no botão *Start* e aguardar a conclusão da execução dos testes.

Durante a execução dos testes no código fonte do projeto informado na seção 1 da ferramenta, os *logs* de arquivos analisados serão exibidos na seção *Analysis*, a barra de progresso indicará o progresso atual e o tempo decorrido na execução do teste e a seção *Security Smells* apresenta os possíveis problemas encontrados durante a análise do projeto.

A Ferramenta fornece como saídas a possibilidade de salvar os *logs* produzidos em arquivo em formato texto e gerar um relatório de Smells, também em

formato texto, a fim de persistir e gerar evidências para sugestão de correções no código ou políticas de permissões, para que o desenvolvedor possa mitigar *Security Smells*.

Está disponível em repositório do GitHub a ferramenta compilada para Microsoft Windows, versão 10 ou superior, em 32 *bits*, o código fonte e elementos necessários para a compilação.

A biblioteca criada para apoiar a ferramenta e definir os Security Smells foi dividida em 2 arquivos no formato JSON, para facilitar a leitura, processamento e futuras revisões e ampliações.

Os *Code Smells*, seção da biblioteca utilizada para analisar o código do projeto testado, foram obtidos a partir da análise de SDKs disponibilizados pelo provedor AWS para as linguagens C#, Java, JS (e Node.js) e PHP. Pode ser carregada na ferramenta via JSON.

Tabela 2: Estrutura da biblioteca de Code Smells

Chave	**Descrição**	**Exemplo**
Language	Linguagem principal usada no projeto	PHP
ARN	Nome do recurso	SES
Cloud	Provedor utilizado	AWS
Method	Método do SDK a ser identificado	SendEmail
Description	Descrição	Envio de e-mail

A Tabela 2 apresenta a estrutura do arquivo JSON da biblioteca de *Code Smells*. A coluna Chave representa a chave do arquivo JSON, a coluna descrição descreve cada chave e, por fim, a coluna Exemplo exemplifica um registro do arquivo com exemplo de valores.

A chave *Language* é utilizada para definir em qual liguaguem o projeto foi principalmente desenvolvido, a chave ARN define qual nome de recurso Amazon está sendo verificado, a chave *Cloud* define qual provedor de nuvem o registro se refere, a chave método apresenta o método a ser buscado no projeto e, por fim, a chave

Description descreve o método.

Os *Cloud Smells*, seção da biblioteca utilizada para analisar os excessos de privilégios contidos nas permissões do usuário IAM do ambriente de produção em Cloud, foram obtidos a partir da análise de documentação e definição de cada permissão disponibilizada pelo provedor AWS. Pode ser carregado na ferramenta via JSON.

Tabela 3: Estrutura da biblioteca de Cloud Smells

Chave	**Descrição**	**Exemplo**
Smell	Smell detectado nas políticas da Cloud	S3:*
ARN	Nome do recurso	S3
Cloud	Provedor utilizado	AWS
Description	Descrição	S3 FullAccess

A Tabela 3 apresenta a estrutura do arquivo JSON da biblioteca de *Cloud Smells*. A coluna Chave representa a chave do arquivo JSON, a coluna descrição descreve cada chave e, por fim, a coluna Exemplo exemplifica um registro do arquivo com exemplo de valores.

A chave *Semell* define qual *smell* deve ser buscado dentro das políticas da *cloud* do usuário testado, a chave ARN define qual nome de recurso Amazon está sendo verificado, a chave *Cloud* define qual provedor de nuvem o registro se refere e, por fim, a chave *Description* descreve o *smell*.

4.2 CONSIDERAÇÕES FINAIS

Este capítulo abordou a ferramenta, sua estrutura, utilização e a biblioteca de *Code Smells* e **Cloud Smells** e sua respectiva estrutura. No capítulo a seguir será apresentada a avaliação experimental da proposta, utilizando a ferramenta desenvolvida em projetos de código aberto.

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo apresenta a avaliação experimental conduzida para avaliar a abordagem proposta com apoio da ferramenta, os testes, a execução dos testes em sete projetos de código aberto, contemplando as linguagens de programação PHP, JS/Node.js e Java. Os testes foram efetuados em um computador x64, utilizando o sistema operacional Windows 10.

5.1 EXPERIMENTO E RESULTADOS

Esta seção apresenta a definição do experimento, a execução do experimento, os projetos selecionados para a avaliação da ferramenta e os resultados em *cloud*.

5.1.1 DEFINIÇÃO DO EXPERIMENTO

Para avaliar a abordagem proposta é importante verificar a assertividade e o tempo necessário para execução dos testes e verificar o comportamento da ferramenta em diferentes linguagens de programação principais e diferentes cenários de configuração do usuário IAM de produção disponibilizado para os testes. Foi conduzida uma avaliação experimental para verificar a abordagem com apoio da ferramenta proposta em 7 projetos de código aberto.

Foram investigadas as seguintes Questões de Pesquisa (QPs):

- **QP1:** Qual a efetividade da abordagem proposta em encontrar *code smells* em aplicações web implementadas em diferentes linguagens de programação, utilizando diferentes ARNs em várias configurações de usuário?

- **QP2:** Qual o comportamento da abordagem considerando o tempo de execução dos testes?

A **QP1** tem como objetivo verificar a efetividade da proposta em ser assertiva buscando *smells* no código da aplicação e no usuário IAM disponibilizado para produção. Foi medida a capacidade da abordagem de encontrar métodos que sugerem uso de recursos em provedores *cloud* nas aplicações testadas, a ausência ou presença de privilégios para a execução desse recurso e a assertividade na detecção dos métodos.

A **QP2** busca, a partir da medição do tempo de execução dos testes, verificar a viabilidade da abordagem na inclusão de mais uma etapa nos testes prévios a publicação ou atualização de versão de uma aplicação *web*.

5.1.2 EXECUÇÃO DO EXPERIMENTO

Para executar o experimento foram utilizados os seguintes recursos de software: Microsoft Windows 10 64bits, Microsoft Visual Studio 2022 e Cloc v1.96.1. Foram selecionados projetos de código aberto, disponíveis no GitHub, com relevância, utlizados no CMS Wordpress, e projetos ativos, com mais de um colaborador.

Tabela 4: Projetos testados

#	Aplicação	Linguagem	Arquivos	Arquivos LP	LOC LP
1	Akaunting	PHP	3877	3065	142161
2	EverShop	NodeJS	1348	500	29755
3	LaraERP	PHP	253	150	3720
4	Plugin AWS SNS	PHP	507	491	29897
5	S3 Plugin	PHP	1517	1187	74727
6	Shopizer	Java	1326	1205	75531
7	Wordpress	PHP	3277	1382	306720

Foram selecionados sete projetos, descritos na Tabela 4, Akaunting[1], EverShop[2], LaraERP[3], Plugin AWS SNS[4], S3 Plugin[5], Shopizer[6] e Wordpress[7] com os respectivos links para o repositório do código fonte, a linguagem utilizada, a quantidade total de arquivos de cada projeto, a quantidade de arquivos na linguagem de programação (LP) principal e a quantidade de linhas de código dos arquivos analisados na linguagem de programação principal do projeto (LOC LP).

Um usuário IAM com nome Mestrado foi criado no provedor de *cloud* AWS, foi configurada uma chave de acesso programático, e atribuída a permissão ***IAMFullAccess*** necessária para execução dos testes. Foram atribuídas políticas *inline*.

Figura 12: Política *S3FullAccess*.

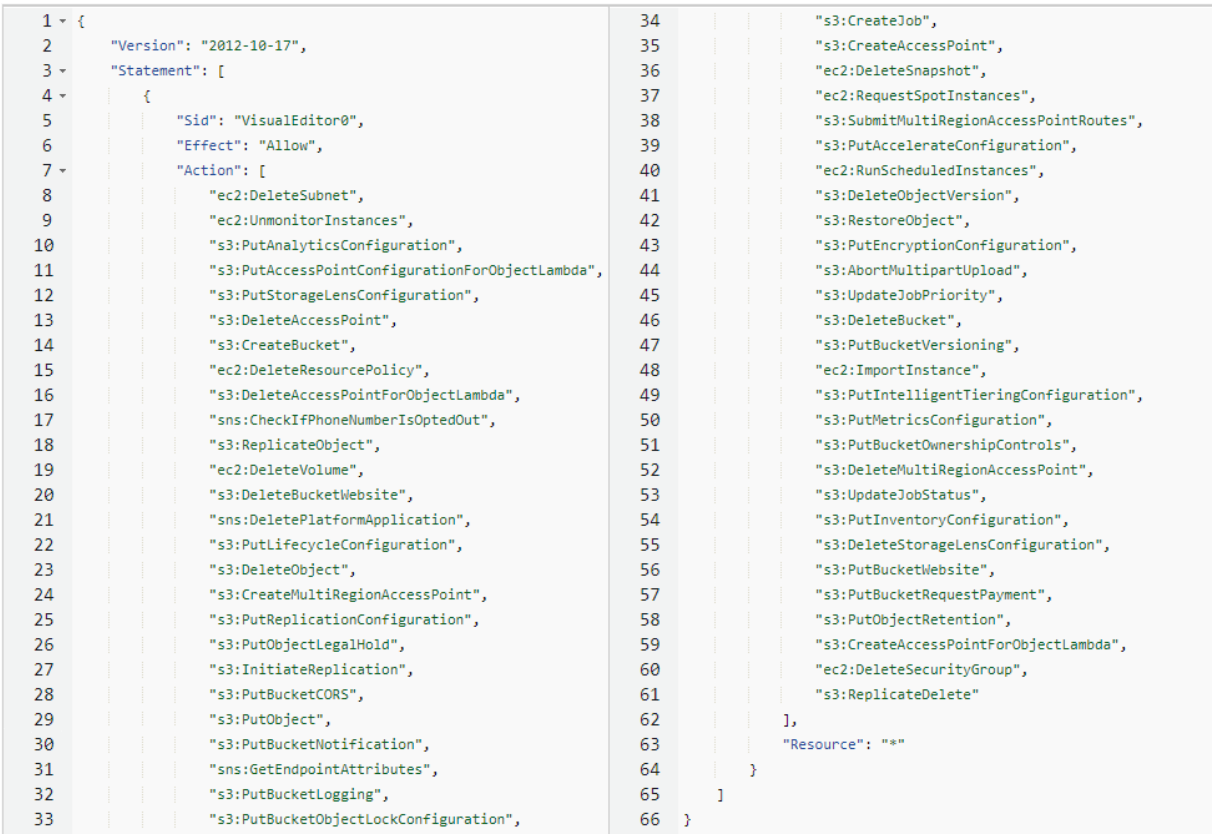
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        }
    ]
}
```

Fonte: O Autor

A Figura 12 apresenta a política nomeada como *S3FullAccess*, descrita *inline*, atribuída ao usuário IAM Mestrado, utilizado para realização dos teste, e descreve a permissão completa, em todos os recursos, ao ARN S3.

[1]Repositório: `https://github.com/akaunting/akaunting`
[2]Repositório: `https://github.com/evershopcommerce/evershop`
[3]Repositório: `https://github.com/laraerp/laraerp`
[4]Repositório: `https://github.com/fayzandotcom/aws-sns-plugin`
[5]Repositório: `https://github.com/deliciousbrains/wp-amazon-s3-and-cloudfront`
[6]Repositório: `https://github.com/shopizer-ecommerce/shopizer`
[7]Repositório: `https://github.com/WordPress/WordPress`

Figura 13: Política *Misc*.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteSubnet",
                "ec2:UnmonitorInstances",
                "s3:PutAnalyticsConfiguration",
                "s3:PutAccessPointConfigurationForObjectLambda",
                "s3:PutStorageLensConfiguration",
                "s3:DeleteAccessPoint",
                "s3:CreateBucket",
                "ec2:DeleteResourcePolicy",
                "s3:DeleteAccessPointForObjectLambda",
                "sns:CheckIfPhoneNumberIsOptedOut",
                "s3:ReplicateObject",
                "ec2:DeleteVolume",
                "s3:DeleteBucketWebsite",
                "sns:DeletePlatformApplication",
                "s3:PutLifecycleConfiguration",
                "s3:DeleteObject",
                "s3:CreateMultiRegionAccessPoint",
                "s3:PutReplicationConfiguration",
                "s3:PutObjectLegalHold",
                "s3:InitiateReplication",
                "s3:PutBucketCORS",
                "s3:PutObject",
                "s3:PutBucketNotification",
                "sns:GetEndpointAttributes",
                "s3:PutBucketLogging",
                "s3:PutBucketObjectLockConfiguration",
                "s3:CreateJob",
                "s3:CreateAccessPoint",
                "ec2:DeleteSnapshot",
                "ec2:RequestSpotInstances",
                "s3:SubmitMultiRegionAccessPointRoutes",
                "s3:PutAccelerateConfiguration",
                "ec2:RunScheduledInstances",
                "s3:DeleteObjectVersion",
                "s3:RestoreObject",
                "s3:PutEncryptionConfiguration",
                "s3:AbortMultipartUpload",
                "s3:UpdateJobPriority",
                "s3:DeleteBucket",
                "s3:PutBucketVersioning",
                "ec2:ImportInstance",
                "s3:PutIntelligentTieringConfiguration",
                "s3:PutMetricsConfiguration",
                "s3:PutBucketOwnershipControls",
                "s3:DeleteMultiRegionAccessPoint",
                "s3:UpdateJobStatus",
                "s3:PutInventoryConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutBucketWebsite",
                "s3:PutBucketRequestPayment",
                "s3:PutObjectRetention",
                "s3:CreateAccessPointForObjectLambda",
                "ec2:DeleteSecurityGroup",
                "s3:ReplicateDelete"
            ],
            "Resource": "*"
        }
    ]
}
```

Fonte: O Autor

A Figura 13 apresenta a política *Misc* atribuída ao usuário IAM Mestrado, e descreve a permissão de diversos métodos, em diferentes ARNs e em todos os recursos.

5.1.3 PROJETOS

Na verificação dos sete projetos selecionados, foram executados testes para encontrar a ocorrência de métodos que sugerem a utilização de recursos em nuvem. Dada a ocorrência, verificou-se junto ao usuário IAM fornecido para o ambiente de produção, a existência de *smells* que causem mau funcionamento da aplicação por ausência de privilégios.

Tabela 5: Resultados dos testes em Projetos

Método		**createBucket**	**createQueue**	**createTopic**	**deleteObject**	**getObject**	**Publish**	**putObject**	**sendEmail**	**sendMessage**	**Subscribe**
ARN		S3	SQS	SNS	S3	S3	SNS	S3	SES	SQS	SNS
Ocorrência		2(100%)	1(100%)	1(100%)	2(100%)	4(100%)	7(100%)	4(100%)	2(100%)	2(100%)	5(100%)
Status		OK	Absent	Absent	OK	Absent	Absent	OK	Absent	Absent	Absent
A1	O	0(0%)	0(0%)	0(0%)	0(0%)	0(0%)	1(14%)	0(0%)	1(50%)	1(50%)	0(0%)
	A	100%	100%	100%	100%	100%	0%	100%	100%	100%	0%
A2	O	0(0%)	0(0%)	0(0%)	0(0%)	0(0%)	1(14%)	0(0%)	0(0%)	0(0%)	1(20%)
	A	100%	100%	100%	100%	100%	0%	100%	100%	100%	0%
A3	O	0(0%)	0(0%)	0(0%)	0(0%)	0(0%)	1(14%)	0(0%)	0(0%)	0(0%)	0(0%)
	A	100%	100%	100%	100%	100%	0%	100%	100%	100%	0%
A4	O	0(0%)	0(0%)	1(100%)	0(0%)	1(25%)	1(14%)	1(25%)	0(0%)	0(0%)	1(20%)
	A	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%

Continua na próxima página

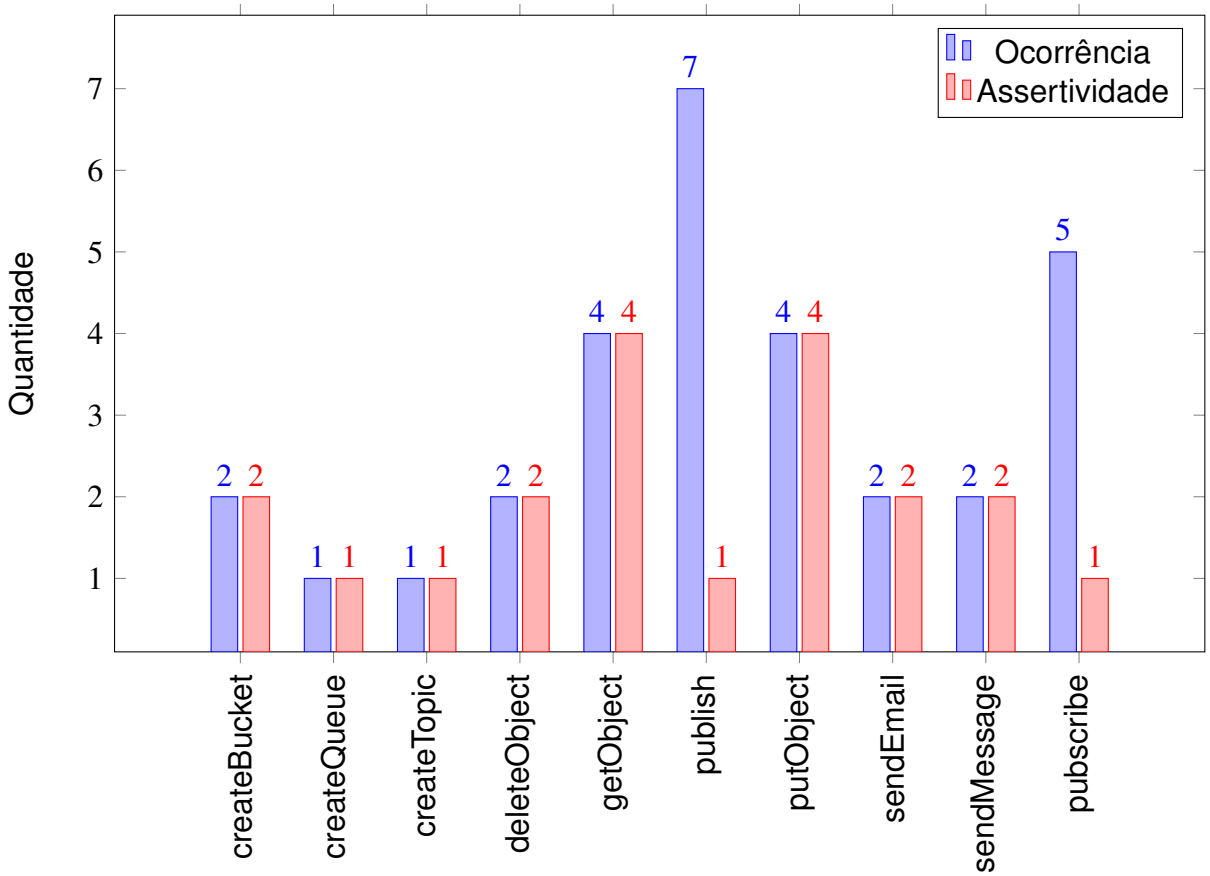
Tabela 5: Resultados dos testes em Projetos (Continuação)

A5	O	1(50%)	0(0%)	0(0%)	1(50%)	1(25%)	1(14%)	1(25%)	0(0%)	1(50%)	1(20%)
	A	100%	100%	100%	100%	100%	0%	100%	100%	100%	0%
A6	O	1(50%)	0(0%)	0(0%)	1(50%)	1(25%)	1(14%)	1(25%)	1(50%)	0(0%)	1(20%)
	A	100%	100%	100%	100%	100%	0%	100%	100%	100%	0%
A7	O	0(0%)	1(100%)	0(0%)	0(0%)	1(25%)	1(14%)	1(25%)	0(0%)	0(0%)	1(20%)
	A	100%	100%	100%	100%	100%	0%	100%	100%	100%	0%
Total Ass.		100%	100%	100%	100%	100%	14,20%	100%	100%	100%	20%

A Tabela 5 apresenta os métodos identificados pela plataforma em cada projeto de aplicação, a ARN que esse método sugere, a ocorrência, o *status*, onde OK representa a existência dessa permissão vinculada a política do usuário IAM em teste, e *Absent* representa a ausência desse privilégio.

Nas linhas A1 a A7, são apresentadas as aplicações, de 1 a 7, respectivamente, Akaunting, EverShop, LaraERP, Plugin AWS SNS, S3 Plugin, Shopizer e Wordpress; com as linhas **O**: indicando a ocorrência ou não do método na aplicação e **A**: indicando a assertividade da ocorrência, ou seja, se a identificação do método pela ferramenta condiz com a utilização do recurso em nuvem, apurado via checagem manual do código fonte.

Figura 14: Gráfico de ocorrência x Assertividade.



Fonte: O Autor

A Figura 14 ilustra a ocorrência do método em aplicações e a assertividade da identificação do método.

5.1.4 CLOUD

A ferramenta verificou a ocorrência de *Smells* na nuvem, independente da aplicação utilizar o recurso, com base na biblioteca, e apontou possíveis más práticas com potencial de gerar problemas de segurança, com isso, ao final dos testes dos projetos, o relatório apresentou os testes realizados na nuvem, com o usuário IAM fornecido.

Tabela 6: Resultados dos testes em Cloud

Smell	**ARN**	**Description**
S3:*	S3	S3 Full Access
S3:DeleteBucket	S3	S3 Delete Bucket
S3:DeleteObjectVersion	S3	S3 Delete Object Version
S3:DeleteAccessPoint	S3	S3 Delete Access Point
S3:DeleteBucketWebsite	S3	S3 Delete Bucket Website
S3:DeleteObject	S3	S3 Delete Object
SNS:GetEndpointAttributes	SNS	SNS Get Endpoint Attributes
SNS:CheckIfPhoneNumberIsOptedOut	SNS	SNS Check If Phone Opted Out
SNS:DeletePlatformApplication	SNS	SNS Delete P. Application
EC2:RunScheduledInstances	EC2	EC2 RunScheduled Instances
EC2:DeleteSubnet	EC2	EC2 Delete Subnet
EC2:DeleteSecurityGroup	EC2	EC2 Delete SecurityGroup
EC2:DeleteResourcePolicy	EC2	EC2 Delete Resource Policy
EC2:DeleteSnapshot	EC2	EC2 Delete Snapshot
EC2:DeleteVolume	EC2	EC2 Delete Volume

A tabela 6 apresenta os *smells* identificados pela ferramenta utilizando o usuário fornecido, que contém as permissões listadas nas Figuras 12 e 13. A assertividade é de 100% em relação aos *smells* dos testes em *cloud* constatados a partir de verificação manual.

QP1: Efetividade

Foi observada a efetividade da abordagem em encontrar *smells* com apoio da ferramenta. Considerando a porcentagem total de assertividade, conforme Tabela 5, observa-se que a abordagem e ferramenta foram capazes de identificar os métodos e associá-los a ARNs, bem como apontar corretamente o *status* de cada método em relação às permissões do usuário IAM fornecido para os testes.

A baixa assertividade relacionada aos métodos *Publish* (14%) e *Subscribe* (20%) se justifica pelo uso frequente destas palavras em outros contextos, por exemplo, o uso do *subscribe* para inscrição em uma lista de e-mails, que não os relacionados às ARNs SNS, do Serviço de Notificação.

QP2: Tempo de Execução

Observou-se um tempo médio de execução dos testes satisfatório, considerando a média de 17 segundos para o menor projeto, AWS SNS, com 29897 linhas de código, conforme Tabela 4, e 172 segundos, em média, para a execução dos testes no projeto mais complexo, com 306720 linhas de código verificadas, conforme Tabela 4. Sendo assim, a sua utilização mostra-se passível de inclusão no processo antes do *deploy* da aplicação.

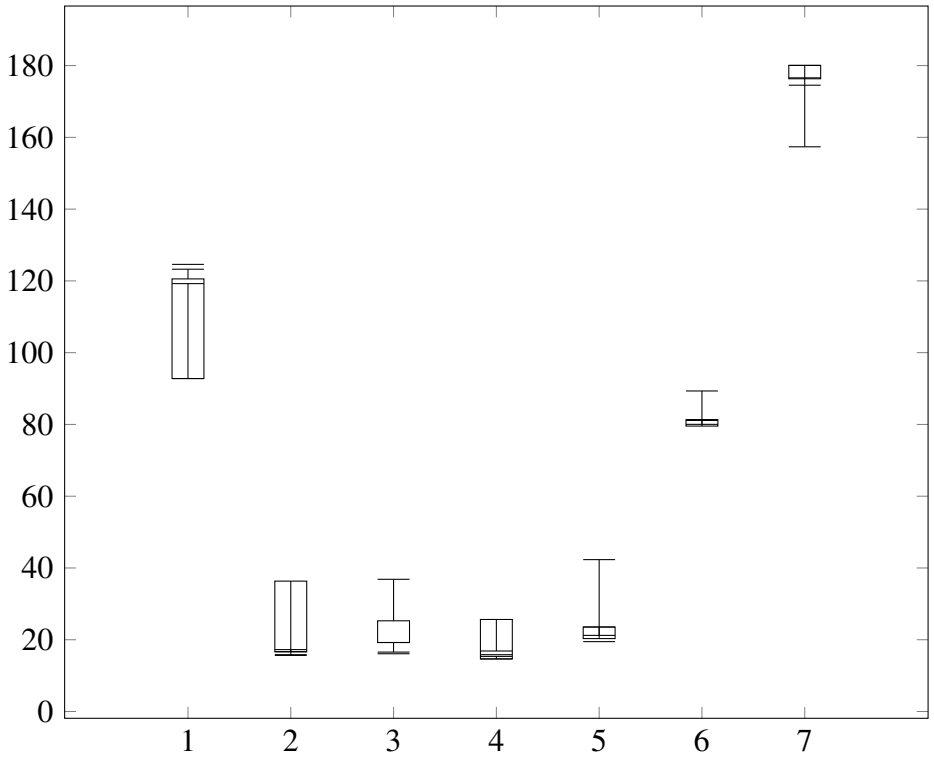
Tabela 7: Tempo de execução dos testes em Projetos

Aplicação	**E1**	**E2**	**E3**	**E4**	**E5**	**Média**
Akaunting	124,6025	92,7854	120,5554	119,2388	123,2548	116,0873
EverShop	15,8369	16,6542	36,3365	17,2541	15,6325	20,3428
LaraERP	16,0935	19,2301	25,2784	16,5561	36,8431	22,8002
AWS SNS	15,8369	14,6533	25,6523	15,3625	16,8632	17,6736
S3 Plugin	19,4727	23,5471	20,3218	21,2125	42,3254	25,3759
Shopizer	81,1083	81,3254	79,5421	80,0257	89,3254	82,2653
Wordpress	174,5323	180,0544	176,3254	157,3665	176,5842	172,9725

A Tabela 7 apresenta, em segundos, o tempo de execução dos testes das sete aplicações testadas, e as colunas E1, E2, E3, E4 e E5 os tempos, em segundos, de

cada uma das cinco execuções realizadas.

Figura 15: Boxplot da variabilidade do tempo de execução em segundos



Fonte: O Autor

A Figura 15 apresenta graficamente a sumarização da distribuição do tempo de execução, os *outliers* são justificados por eventuais variações no tempo de resposta da internet e disponibilidade do disco na máquina utilizada nos testes.

5.2 DISCUSSÃO DOS RESULTADOS

A avaliação experimental verificou a viabilidade, aplicabilidade e assertividade da abordagem e ferramenta propostas, com foco em aplicações web em versões de ambiente de produção.

Com os resultados apresentados, notou-se a aplicabilidade da abordagem e utilização da ferramenta, oferecendo maior segurança e verificação antecipada na identificação de possível problemas de permissão.

A Linguagem de programação principal utilizada no projeto a ser testado

não interferiu na eficiência da ferramenta nem apresentou aumento na dificuldade de encontrar *smells*.

5.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a avaliação experimental da abordagem com apoio da ferramenta, o experimento, sua definição, execução e resultados em projetos e na *cloud*, bem como a apresentação e resposta de duas questões de pesquisa. O próximo capítulo conclui este trabalho.

6 CONCLUSÃO

Este trabalho apresentou duas contribuições na área de segurança e implementação de aplicações *web* em provedores de nuvem, a primeira foi a identificação de um número considerável de respostas que, embora corretas a princípio, geram problemas de segurança futuros para a aplicação em fase de produção, bem avaliadas e recomendadas em fóruns utilizados frequentemente por desenvolvedores inexperientes e experientes para a solução de problemas pontuais relacionados a permissões e políticas de acesso de provedores *cloud*. Esta contribuição foi possível a partir do desenvolvimento, execução e análise dos dados pela ferramenta de mineração, apresentada em 6.2.1.

A segunda contribuição foi o desenvolvimento da ferramenta CloudSSF, que instancia a abordagem proposta e oferece a desenvolvedores a oportunidade de verificar, antes de disponibilizar uma aplicação web em ambiente de produção, se as políticas concedidas ao usuário IAM de ambiente de produção apresentam ausência ou excesso de privilégios que causam mau funcionamento ou possíveis problemas de segurança.

6.1 LIMITAÇÕES E TRABALHOS FUTUROS

A seguir estão descritas as limitações do projeto e sugestões para melhorias em trabalhos futuros. Foram identificadas as seguintes limitações e apresentadas as sugestões:

- Comentários não são ignorados na busca da ferramenta, o que pode gerar falsos positivos;

- A utilização de um único provedor limita a abrangência do estudo e induz vícios de utilização devido a experiência do pesquisador.

Ao longo da condução do trabalho, algumas melhorias surgiram, com isso, sugere-se para continuidade da pesquisa e trabalhos futuros:

- a ampliação da biblioteca e adição de novas linguagens de programação cobertas pela ferramenta;
- a inclusão de novos provedores *cloud*, além da AWS;
- reescrita da ferramenta em linguagem multiplataforma para possibilitar o uso em outros sistemas operacionais além do Microssoft Windows;
- inclusão de diretórios ignorados na análise do código;
- inclusão de filtro de linguagem de programação a ser analisada;
- automatizar a identificação da linguagem de programação do projeto analisado;
- possibilidade de analisar projetos multiliguagem, (e.g. Node e PHP em um mesmo projeto Laravel);
- desenvolvimento de um guia com diretrizes de segurança para nortear usuários com pouca experiência no processo de publicação de aplicações em ambiente de nuvem.

6.2 DIVULGAÇÃO DOS RESULTADOS

Durante o desenvolvimento desta dissertação de mestrado foi desenvolvida a ferramenta ClodSSF, com registro requerido junto ao INPI e encontra-se publicada no GitHub do autor, junto ao pacote experimental.

6.2.1 FERRAMENTAS DESENVOLVIDAS E PACOTE EXPERIMENTAL

- DEIZEPE, Anderson Roberto; SCANNAVINO, Katia Romero Felizardo. Mineração de dados utilizado Google em fórum na internet. Disponível em: https://github.com/Deizepe/MestradoScriptPyStackOverflow

- DEIZEPE, Anderson Roberto. CloudSSF - Cloud Security Smell Finder. Disponível em: https://github.com/Deizepe/CloudSSF

6.2.2 REGISTRO DE SOFTWARE

- DEIZEPE, Anderson Roberto. CloudSSF - Cloud Security Smell Finder. Data da Publicação: 07/08/2023. Certificado de Registro de Programa de Computador, Registro N° BR512023002303-5

REFERÊNCIAS

ACAR, Y.; BACKES, M.; FAHL, S.; KIM, D.; MAZUREK, M. L.; STRANSKY, C. You Get Where You're Looking for: The Impact of Information Sources on Code Security. **Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016**, p. 289–305, 2016.

AKAMAI. **Segurança de App e API**. 2021. Disponível em: <https://www.akamai.com/pt/solutions/security/app-and-api-security>.

ALIBABA. **Alibaba Cloud**. 2021. Disponível em: <https://us.alibabacloud.com/>.

ALMORSY, M.; GRUNDY, J.; MüLLER, I. **An Analysis of the Cloud Computing Security Problem**. 2016.

APACHE. **Kafka 2.8 Documentation**. 2021. Disponível em: <https://kafka.apache.org/documentation/>.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. A view of cloud computing. **Communications of the ACM**, v. 53, n. 4, p. 50–58, 2010. ISSN 00010782.

BUYYA, R.; BROBERG, J.; GOSCINSKI, A. **Cloud Computing: Principles and Paradigms**. Wiley, 2010. (Wiley Series on Parallel and Distributed Computing). ISBN 9781118002209. Disponível em: <https://books.google.com.br/books?id=S1NvRRd77rQC>.

CHEN, Y.; PAXSON, V.; KATZ, R. H. **What's New About Cloud Computing Security?** [S.l.], 2010. Disponível em: <https://www2.eecs.berkeley.edu/pubs/techrpts/2010/eecs-2010-5.html>.

DILLON, T.; WU, C.; CHANG, E. Cloud computing: Issues and challenges. In: **2010 24th IEEE International Conference on Advanced Information Networking and Applications**. [S.l.: s.n.], 2010. p. 27–33.

FOWLER, M. **Refactoring: Improving the Design of Existing Code**. USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0201485672.

FOWLER, M.; BECK, K.; BRANT, J.; OPDYKE, W.; ROBERTS, D. **Refactoring: Improving the Design of Existing Code**. USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0201485672.

GARTNER. **Gartner Forecasts Worldwide Security and Risk Management Spending to Exceed $150 Billion in 2021**. [S.l.], 2021. Disponível em: <https://www.gartner.com/en/newsroom/press-releases/2021-05-17-gartner-forecasts-worldwide-security-and-risk-managem>.

GITHUB. **Rank of top languages on github.com**. 2021. Disponível em: <https://madnight.github.io/githut/pull_requests/2022/1>.

GNU. **A Licença Pública Geral GNU GPL**. 2021. Disponível em: <https://www.gnu.org/licenses/licenses.html>.

GOOGLE. **Google Cloud Platform**. 2021. Disponível em: <https://cloud.google.com/>.

IBM. **IBM Cloud**. 2021. Disponível em: <https://cloud.ibm.com/>.

IYER, P.; MASOUMZADEH, A. Mining Positive and Negative Attribute-Based Access Control Policy Rules. In: **Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies**. New York, NY, USA: ACM, 2018. p. 161–172. ISBN 9781450356664. Disponível em: <https://dl.acm.org/doi/10.1145/3205977.3205988>.

MARSTON, S.; LI, Z.; BANDYOPADHYAY, S.; ZHANG, J.; GHALSASI, A. Cloud computing — the business perspective. **Decision Support Systems**, v. 51, n. 1, p. 176–189, 2011. ISSN 0167-9236. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167923610002393>.

MELL, P.; GRANCE, T.; GRANCE, T. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. 2011.

MICROSOFT. **Microsft Azure**. 2021. Disponível em: <https://azure.microsoft.com/pt-br/>.

RAHMAN, A.; PARNIN, C.; WILLIAMS, L. The seven sins: Security smells in infrastructure as code scripts. In: **2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)**. [S.l.: s.n.], 2019. p. 164–175.

RITTINGHOUSE, J. W.; RANSOME, J. F. **Cloud Computing: Implementation, Management and Security**. [S.l.]: CRC Press, 2016. ISBN 978-1439806807.

RODRIGUES, L. **Hackers invadem servidores da Amazon para minerar Monero**. 2020. Disponível em: <https://www.criptofacil.com/hackers-invadem-servidores-amazon-para-minerar-monero/>.

SABIR, F.; PALMA, F.; RASOOL, G.; GUéHéNEUC, Y.-G.; MOHA, N. A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems. **Software: Practice and Experience**, John Wiley and Sons Ltd, v. 49, p. 3–39, 1 2019. ISSN 00380644. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1002/spe.2639>.

SAEED, I.; BARAS, S.; HAJJDIAB, H. Security and Privacy of AWS S3 and Azure Blob Storage Services. In: **2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)**. IEEE, 2019. p. 388–394. ISBN 978-1-7281-1322-7. Disponível em: <https://ieeexplore.ieee.org/document/8821735/>.

SANDERS, M. W.; YUE, C. Mining least privilege attribute based access control policies. In: **Proceedings of the 35th Annual Computer Security Applications Conference**. New York, NY, USA: ACM, 2019. p. 404–416. ISBN 9781450376280. Disponível em: <https://dl.acm.org/doi/10.1145/3359789.3359805>.

SERVICES, A. W. **Amazon Simple Email Service: Developer Guide**. Amazon Web Services, 2021. 915 p. Disponível em: <https://www.amazon.com.br/Amazon-Simple-Email-Service-Developer-ebook/dp/B07646TW8P>.

SERVICES, A. W. **Amazon Simple Notification Service: Developer Guide**. [S.l.]: Amazon Web Services, 2021. 427 p.

SERVICES, A. W. **Amazon Simple Queue Service: Developer Guide**. Amazon Web Services, 2021. 318 p. Disponível em: <https://www.amazon.com.br/Amazon-Simple-Queue-Service-Developer-ebook/dp/B07644Y214/>.

SERVICES, A. W. **Amazon Simple Storage Service: Developer Guide**. Amazon Web Services, 2021. 1504 p. Disponível em: <https://www.amazon.com.br/Amazon-Simple-Storage-Service-Developer-ebook/dp/B0763Z4YVV>.

SERVICES, A. W. **AWS Identity and Access Management: User Guide**. Amazon Web Services, 2021. 3020 p. Disponível em: <https://www.amazon.com.br/AWS-Identity-Access-Management-English-ebook/dp/B07642VLTV>.

SERVICES, A. W. **O que é a computação em nuvem?** 2021. Disponível em: <https://aws.amazon.com/pt/what-is-cloud-computing/>.

SHARMA, D. H.; DHOTE, C. A.; POTEY, M. M. Identity and Access Management as Security-as-a-Service from Clouds. **Procedia Computer Science**, Elsevier Masson SAS, v. 79, p. 170–174, 2016. ISSN 18770509. Disponível em: <https://dx.doi.org/10.1016/j.procs.2016.03.117>.

SILVA, C. M. R. da; SILVA, J. L. C. da; RODRIGUES, R. B.; NASCIMENTO, L. M. do; GARCIA, V. C. Systematic mapping study on security threats in cloud computing. **IJCSIS) International Journal of Computer Science and Information Security**, v. 11, 3 2013. Disponível em: <https://arxiv.org/abs/1303.6782>.

SILVA, P. Rodrigo da; SANTOS, V.; SOUZA, E.; MEINERZ, G.; FELIZARDO, K.; VIJAYKUMAR, N. Extraction of useful information from unstructured data in software engineering: A systematic mapping. In: . [S.l.: s.n.], 2020.

SJOBERG, D. I.; YAMASHITA, A.; ANDA, B. C.; MOCKUS, A.; DYBå, T. Quantifying the effect of code smells on maintenance effort. **IEEE Transactions on Software Engineering**, v. 39, n. 8, p. 1144–1156, 2013.

SURYANARAYANA, G.; GANESH, S.; SHARMA, T. **Refactoring for Software Design Smells: Managing Technical Debt**. [S.l.: s.n.], 2014. 1-237 p. ISBN 978-0128013977.

Synergy Research Group. **Cloud Market Ends 2020 on a High while Microsoft Continues to Gain Ground on Amazon**. 2021. 1 p. Disponível em: <https://www.srgresearch.com/articles/cloud-market-ends-2020-high-while-microsoft-continues-gain-ground-amazon>.

Synergy Research Group. **Quarterly Cloud Market Leaps to 42B – Amazon, Microsoft & Google Pocket 63 percent of Dollars Spent**. [S.l.], 2021. Disponível em: <https://www.srgresearch.com/articles/quarterly-cloud-market-leaps-to-42b-amazon-microsoft-google-pocket-63-of-dollars-spent>.

VAQUERO, L. M.; RODERO-MERINO, L.; CACERES, J.; LINDNER, M. A break in the clouds. **ACM SIGCOMM Computer Communication Review**, v. 39, n. 1, p. 50–55, 2008. ISSN 0146-4833.

YAMASHITA, A.; MOONEN, L. Do developers care about code smells? an exploratory survey. In: **2013 20th Working Conference on Reverse Engineering (WCRE)**. [S.l.: s.n.], 2013. p. 242–251.

ZISSIS, D.; LEKKAS, D. Addressing cloud computing security issues. **Future Generation Computer Systems**, v. 28, n. 3, p. 583–592, 2012. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X10002554>.

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

PPGI
programa de pós-graduação
em informática

@ppgicp

www ppgi-cp@utfpr.edu.br
portal.utfpr.edu.br/cp/ppgi