Universidade Federal de Juiz de Fora
Instituto de Ciência Exatas
Bacharelado em Ciência da Computação

Segurança de APIs em Sistemas em Nuvem

Ronaldo Modesto Ponciano

JUIZ DE FORA
DEZEMBRO, 2023

Segurança de APIs em Sistemas em Nuvem

Ronaldo Modesto Ponciano

Universidade Federal de Juiz de Fora
Instituto de Ciência Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Eduardo Pagani Julio

JUIZ DE FORA
DEZEMBRO, 2023

Segurança de APIs em Sistemas em Nuvem

Ronaldo Modesto Ponciano

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIA EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Eduardo Pagani Julio
Dr

José Maria Nazar David
Dr

Luciano Jerez Chaves
Dr

JUIZ DE FORA
12 DE DEZEMBRO, 2023

A Deus que me deu forças para vencer esse grande desafio.

A minha mãe por ter me criado e me ensidado a ser uma boa pessoa, além de me ajudar na dura caminhada que foi minha graduação.

Aos meus irmãos e amigos.

Resumo

O período de 2019 a 2021 foi impactado pela pandemia de COVID-19, resultando em muitas perdas de vida até a descoberta de vacinas. Com a necessidade de isolamento, empresas e pessoas tiveram que se adaptar ao trabalho remoto e ao aumento do uso de sistemas computacionais, o que resultou no aumento do volume de dados sensíveis em tráfego dado que esses dados passaram a ser cada vez mais necessários para que esses sistemas pudessem funcionar. Neste trabalho, é realizado um estudo exploratório em várias APIs, utilizando algumas ferramentas de pentest juntamente com metodologias de testes fornecidas pela OWASP para detectar determinados tipos de falhas.

Os testes revelaram que grande parte das APIs analisadas apresenta pelo menos uma falha, sendo a grande maioria das falhas simples de encontrar e corrigir. Isso indica que a qualidade das APIs pode estar caindo consideravelmente. Com os resultados obtidos, são levantadas algumas ações que podem ser tomadas para aumentar a qualidade das APIs e sistemas web.

Palavras-chave: Cibersegurança, Nuvem, API.

Abstract

The period from 2019 to 2021 was impacted by the COVID-19 pandemic, resulting in significant loss of life until the discovery of vaccines. With the need for isolation, businesses and individuals had to adapt to remote work and increased usage of computer systems, leading to a surge in sensitive data traffic as these data became crucial for these systems to function.

This study conducts an exploratory analysis on multiple APIs, employing certain pentesting tools along with testing methodologies provided by OWASP to detect specific types of vulnerabilities.

The tests revealed that a large portion of the analyzed APIs exhibits at least one vulnerability, with the majority being relatively simple to detect and rectify. This suggests a potential decline in API quality. Based on the findings, several actions are proposed to enhance the quality of APIs and web systems.

Keywords: Cybersecurity, cloud, API.

Agradecimentos

Ao professor Eduardo Pagani Julio pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos demais professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"Lembra que o sono é sagrado e alimenta de horizontes o tempo acordado de viver".

Beto Guedes (Amor de Índio)

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

l l

DCC Departamento de Ciência da Computução

UFJF Universidade Federal de Juiz de Fora

OWASP Open Web Application Security Project

ISP Internet Service Provider

API Application Program Interface

CI/CD Continuous Integration/Continuous Delivery

DEVOPS Dev (desenvolvimento) e Ops (operações)

SOAP Simple Object Access Protocol

XML Extensible Markup Language

REST Representational State Transfer

WIFI Wireless Fidelity

IAAS Infrastructure as a Service (infraestrutura como serviço)

PAAS Plataform as a Service (plataforma como serviço)

SAAS Software as a Service (software como serviço)

SO Sistemas Operacional

URI Uniform Resource Identifier

GraphQL Graph Query Language

RPC Remote Procedure Call

gRPC Google Remote Procedure Call

CSP Cloud Service Provider (Provedor de serviços em Nuvem)

VM Virtual Machine (Máquina virtual)

CPU Central Process Unity (Unidade de processamento central)

XSS Cross-site Scripting

PENTEST Penetration Test (Teste de intrusão)

CWE Common Weakness Enumeration

Go Golang(Linguagem de programação)

Rubby Linguagem de programação

MacOs Macintosh Os

MIT Massachusetts Institute of Technology

GLP General Public License

GUI Graphic User Interface

Android Sistema Operacional Android

URL Uniform Resource Locator

SQL Structured Query Language

HTTP Hipertext Transfer Protocol

HTTPS Hipertext Transfer Protocol Secure

IA Inteligência Artificial

1 Introdução

O uso de sistemas cresceu consideravelmente entre os anos de 2019 e 2020 devido à necessidade de isolamento por parte da população para evitar a disseminação do vírus da Covid-19 (CIOTTI et al., 2020), principalmente no primeiro ano da pandemia, quando ainda não existia nenhum tipo de vacina.

Dado o aumento exponencial do volume de operações em sistemas online, as grandes empresas e fabricantes de software tiveram que adotar novos paradigmas de programação para atender à crescente demanda por novas funcionalidades, ao mesmo tempo em que melhoravam os serviços já existentes. Um desses paradigmas é o desenvolvimento de APIs e microserviços (*micro services*) (THÖNES, 2015) em nuvem, uma abordagem que já estava ganhando espaço e foi potencializada durante a pandemia

APIs são sistemas com uma interface para interagir com outros sistemas e/ou usuários. Normalmente utilizam o padrão *REST* (PETRILLO et al., 2016), mas outros padrões também são usados como, por exemplo, *SOAP* (BOX et al., 2000) ou *XML* (YERGEAU et al., 2004). Estas características tornam as APIs uma solução altamente escalável, e por isso são escolhidas por diversas empresas, especialmente quando combinadas com o ambiente em nuvem. No entanto, essa combinação também tem suas desvantagens.

1.1 O Problema

A grande demanda por sistemas computacionais resultou em um aumento de informações sensíveis na Internet como por exemplo números de cartões de crédito, documentos pessoais e dados de saúde, presentes em vários bancos de dados. No entanto, muitas empresas não tomam as devidas precauções para proteger essas informações, o que por vezes acarreta em vazamentos de dados sensíveis dos usuários ou até mesmo das próprias empresas.

A pandemia acelerou a corrida pelo desenvolvimento de software nas empresas, com a necessidade de lançar soluções rapidamente para evitar a concorrência. Isso resultou em mudanças no processo de desenvolvimento, com o uso mais frequente de técnicas como

CI/ID (JOHNSTON, 2020), *DEVOPS* (EBERT et al., 2016) e *SCRUM* (SCHWABER, 1997) para aumentar a velocidade de produção dos sistemas por meio de iterações curtas (Sprints) no ciclo de desenvolvimento, proporcionando entregas frequentes. A priorização de tarefas (Backlog) traz maior organização e visibilidade do que precisa ser desenvolvido. Além disso, há transparência e comunicação por meio de reuniões diárias, chamadas Daily, nas quais os desenvolvedores relatam em qual tarefa trabalharam no último dia de trabalho anterior à reunião, qual tarefa irão fazer após a reunião e se possuem algum impedimento. Isso ajuda a prevenir que algum desenvolvedor fique impedido por falta de recursos ou qualquer outra razão, o que poderia causar atrasos no desenvolvimento do software.

No entanto, essa abordagem também tem suas desvantagens, por exemplo:

- **Complexidade de Implementação:** Implementar o Scrum pode exigir uma mudança significativa na cultura organizacional e nos processos existentes, o que pode ser complexo e demorado.
- **Necessidade de Engajamento Total:** O sucesso do Scrum depende do comprometimento total de toda a equipe. Se alguns membros não estiverem totalmente engajados ou se a liderança não apoiar adequadamente, o framework pode não ser tão eficaz.
- **Exige Profundo Conhecimento:** Para aproveitar ao máximo o Scrum, a equipe precisa ter um entendimento profundo do framework e de seus princípios. A falta desse conhecimento pode levar a implementações ineficazes.
- **Rigidez para Mudanças Fora das Sprints:** Mudanças imprevistas fora das Sprints podem ser difíceis de lidar no contexto do Scrum, já que o framework valoriza a estabilidade durante cada ciclo.
- **Foco Exclusivo na Entrega de Software Funcional:** O Scrum se concentra na entrega contínua de software funcional, o que pode negligenciar a atenção a outros aspectos importantes, como testes de segurança ou manutenção.
- **Ênfase na Quantidade em Detrimento da Qualidade:** Em alguns casos, a ênfase na entrega frequente pode levar a uma redução na qualidade do software,

especialmente se os processos de teste e revisão forem comprometidos para atender aos prazos das Sprints.

- **Dificuldade de Estimativas Precisas:** Estimar o tempo necessário para completar as tarefas pode ser desafiador, o que pode levar a problemas de planejamento e atrasos.

- **Desafios na Escalabilidade:** Escalar o Scrum para equipes grandes ou projetos complexos pode ser complicado e exigir adaptações que nem sempre são tão eficazes quanto o modelo original.

O uso de ambientes na nuvem possui suas limitações, muitas vezes não levadas em consideração ou mesmo desconhecidas. Um entendimento errôneo de como esse ambiente funciona pode levar a uma perda de segurança nos sistemas, pois, apesar de algumas brechas presentes em sistemas monolíticos serem automaticamente resolvidas com a utilização da nuvem, existem novas falhas, cenários, limitações e paradigmas que formam um ambiente ainda pouco conhecido, assim como as vulnerabilidades que o permeiam.

Outra mudança que já estava em curso, e que foi intensificada devido à pandemia da COVID-19, foi a implementação do *home office*. Essa forma de trabalho também representa maiores riscos para as empresas, pois, na maioria das vezes, os funcionários estão em uma rede WIFI doméstica, em hotéis ou aeroportos. Essas redes são perigosas, pois não oferecem o mesmo nível de segurança de uma rede privada de um escritório, por exemplo. Por fim, é preciso considerar, ainda, os equipamentos de trabalho dos profissionais que muitas vezes é pessoal o que também gera riscos pois sendo algo pessoal a empresa não tem controle sobre o que é instalado.

1.2 Contextualização

O conhecimento que se tem atualmente sobre como o ambiente da nuvem funciona ainda é muito limitado, diversas empresas buscam realizar pesquisas com o objetivo de obter um melhor entendimento de como esse meio funciona e como é possível obter maiores níveis de segurança dos ativos (recursos computacionais, dados sensíveis e dos usuários por exemplo) que se utilizam desse modelo de arquitetura.

1.3 Justificativa/Motivação

Para garantir a segurança das informações que são trafegadas diante do novo cenário potencializado pela pandemia da COVID-19, é necessário o uso de novas ferramentas e técnicas para proteger esses ativos. É preciso repensar a forma como são desenvolvidas as APIs, como esses sistemas são gerenciados no ambiente da nuvem e como controlar os ambientes de desenvolvimento utilizados no dia a dia dos desenvolvedores. É preciso revisar as técnicas e ferramentas utilizadas na cibersegurança (*Cyber security*) para elevar o nível de segurança dos sistemas que tendem a utilizar cada vez mais uma infraestrutura na nuvem.

1.4 Objetivos

A obra pode ser dividida em objetivos gerais e específicos.

Como objetivos gerais, este trabalho se propõe a demonstrar que o ambiente da nuvem possui falhas que podem afetar as APIs que nele estão hospedadas, mostrando através de exemplos que esse ambiente não oferece segurança por padrão, e que os arquitetos precisam levar em consideração vários aspectos de segurança ao projetar seus sistemas.

O estudo tem como objetivos específicos identificar as falhas mais comuns nas APIs por meio da metodologia de teste da OWASP WSTG, bem como propor estratégias para lidar com esses pontos fracos, visando fortalecer a segurança e confiabilidade das APIs e sistemas web. Além disso, é objetivo dessa obra sugerir ferramentas que podem ser utilizadas para detectar as vulnerabilidades encontradas, bem como ações que podem ser tomadas para reforçar a segurança das APIs.

1.5 Metodologia

A pesquisa desenvolvida aqui irá mesclar duas metodologias e tem como objetivo demonstrar e discutir falhas que podem ocorrer ao se utilizar um ambiente de computação na nuvem (e até mesmo ambientes que não são da nuvem).

Além disso, será apresentado um panorama de como está a segurança de APIs publicadas na internet. Para esse cenário, iremos utilizar a abordagem de pesquisa exploratória, tanto manualmente quanto com o auxílio de algumas ferramentas.

Nesta parte, a pesquisa será limitada a ir apenas até a fase de detecção de vulnerabilidades, dado que explorar vulnerabilidades em sistemas para os quais não se tem permissão é contra a lei.

De forma geral, pode-se definir a estrutura da pesquisa da seguinte forma:

- **Levantamento de ferramentas**: nesta etapa será feito o levantamento das ferramentas que serão utilizadas. Para isso, será feita uma lista de possíveis ferramentas que sejam capazes de detectar falhas em APIs, sejam elas cloud ou não.

- **Levantamento dos cenários de teste**: em seguida, serão listados os cenários de teste que serão submetidos às APIs. Serão considerados relatórios de falhas recentes, bem como falhas já relatadas nos artigos analisados no Capítulo 3 (Trabalhos relacionados).

- **Levantamento dos alvos**: nesta fase serão selecionadas as APIs e sistemas alvo.

- **Montagem do plano de testes**: será definido o plano de testes levando em consideração as características dos provedores de serviço em nuvem e as ferramentas escolhidas, bem como os casos de teste levantados. Essa análise permitirá elaborar um plano de testes que possa testar o maior número possível de situações;

- **Execução dos testes e coleta dos resultados**: os testes serão realizados e os resultados serão obtidos e compilados. Nesta fase serão evidenciados quais foram as falhas que mais ocorreram e quantas vezes cada uma ocorreu.

- **Compilação dos resultados**: por fim, os resultados obtidos serão analisados e compilados, bem como serão indicadas ferramentas que podem ser utilizadas e ação que podem ser tomadas para mitigar as falhas detectadas durante o estudo.

Sob nenhuma hipótese será realizada a exploração de uma falha, pois não é objetivo desse estudo danificar ou prejudicar, de qualquer forma, sistemas e APIs de empresas privadas e/ou públicas

2 Fundamentação Teórica

Para que seja possível enxergar os pontos de falhas que as APIs em Nuvem podem esconder, é necessário compreender a estrutura de uma arquitetura desenvolvida utilizando esse ambiente, seus componentes, conceitos relacionados, tecnologias empregadas e, principalmente, como essas coisas se somam para formar o que hoje é conhecido como Nuvem.

Por isso, neste capítulo serão apresentados os conceitos que definem esse ambiente, perpassando por suas estruturas e implementações que possibilitam que toda essa estrutura funcione de forma a permitir abrigar os mais variados tipos de aplicações, bem como serão definidos termos e conceitos importantes para o entendimento do conteúdo desta obra.

Por fim, será feita uma introdução ao tema de cibersegurança. Isso servirá como base para um estudo mais profundo que será desenvolvido ao longo do trabalho.

2.1 O que é uma API?

Dado que as APIs serão o foco desse trabalho, se faz necessário definir e compreender melhor o que elas são e como funcionam.

Jacobson, Brail e Woods (2012) define API como "interface de programação de aplicativos". Uma API pode fornecer uma ligação para parceiros ou desenvolvedores de terceiros para acessar dados e serviços para criar aplicações. As APIs do Twitter e do Facebook são famosos exemplos. Existem APIs abertas a qualquer desenvolvedor, APIs abertas apenas a parceiros, APIs que são usadas internamente para facilitar a colaboração entre as equipes."

A Figura 2.1 ilustra a estrutura básica de uma API.

2.1.1 Tipos de APIs

Segundo Qian et al. (2009) existem 2 tipos de APIs. As públicas e as privadas. As públicas são acessíveis por qualquer um na internet, eventualmente elas exigirão alguma forma de

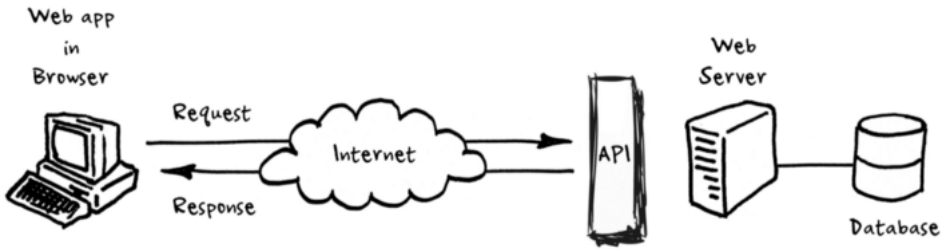

Figura 2.1: Exemplo que ilustra de forma resumida o que é uma API *Fonte*: Google

autenticação (como por exemplo um cadastro), geralmente não é necessário pagar para utilizar os seus recursos.

Já as APIs privadas são acessadas apenas por organizações privadas, não sendo abertas ao público e, normalmente, estão sob uma forte camada de segurança, com políticas de acesso muito bem definidas.

2.1.2 Padrões e protocolos utilizados em APIs

Existem vários padrões e protocolos utilizados na construção de APIs atualmente. É importante conhecê-los pois cada padrão trata os dados recebidos de forma distinta, o que constantemente é explorado por invasores que visam denegrir o estado do serviço e prejudicar o funcionamento do sistema. Os três mais utilizados são REST, SOAP, GraphQL. Seguem breves definições de cada um desses padrões, segundo RedhatC (2020);

- **REST**: é um acrônimo para *Representational State Transfer*. REST é um estilo arquitetônico. A premissa básica é que os desenvolvedores utilizem os métodos HTTP padrão, como por exemplo GET, POST, PUT e DELETE, para consultar e modificar recursos representados por URIs na internet;

- **SOAP**: *Simple Object Access Protocol* é um protocolo para troca de informações codificadas em Extensible Markup Language *Extensible Markup Language* (XML) entre um cliente e um serviço ou procedimento residente na Internet. A especificação foi tornada pública em 1999 pelo W3C e é um padrão aberto.

- **GraphQL**: é uma tecnologia de código aberto criada pelo Facebook. GraphQL é uma especificação independente de linguagem. Há muitas implementações da

especificação em uma variedade de linguagens de programação, como por exemplo Go, .NET/C, Node.js, Python, entre outras;

- **gRPC**: é uma tecnologia de troca de dados desenvolvida pelo Google e posteriormente tornada de código aberto. Assim como o GraphQL é uma especificação implementada em várias linguagens de programação. A diferença é que ao contrário do REST e do GraphQL que utilizam formatos de dados baseados em texto, o gRPC usa o formato binário. No entanto, o uso deste protocolo exige que tanto o cliente quanto o servidor na troca de dados gRPC tenham acesso à mesma definição do formato dos dados. O gRPC utiliza o formato de dados binários por algumas razões, a saber:
  - **Eficiência**: os dados binários são mais compactos e, portanto, mais eficientes em termos de uso de largura de banda e armazenamento do que formatos de texto, como JSON ou XML. Isso é crucial, especialmente em sistemas distribuídos e comunicações de rede, onde a otimização de recursos é essencial.
  - **Velocidade**: o processamento de dados binários é mais rápido do que o processamento de texto. Ao usar buffers, o gRPC pode oferecer uma comunicação mais rápida entre os serviços, resultando em tempos de resposta mais ágeis.
  - **Interoperabilidade e linguagens de programação**: os dados binários são mais facilmente interpretados por várias linguagens de programação. O gRPC é uma estrutura que permite a comunicação entre serviços escritos em diferentes linguagens, e os dados binários são mais simples de serem interpretados de forma consistente em diversas plataformas.
  - **Suporte a tipos de dados complexos**: estruturas de dados complexas podem ser representadas de maneira mais eficiente em formato binário do que em texto. Isso é particularmente útil quando se lidam com dados complexos ou estruturas de mensagem em sistemas distribuídos.

2.2 Computação em Nuvem

Para realizar uma análise do ambiente de computação na Nuvem e identificar possíveis vulnerabilidades de segurança, é fundamental compreender o que é esse ambiente bem como sua estrutura, componentes e conceitos subjacentes a esse paradigma.

2.2.1 O conceito de Nuvem

Atualmente o termo Nuvem (Cloud) possui muitas definições.

Segundo Qian et al. (2009), o conceito de Nuvem pode ser assim definido: "É um tipo de técnica de computação em que os serviços de TI são fornecidos por grandes unidades de computação de baixo custo conectadas por redes IP. A computação em nuvem está enraizada no design da plataforma do mecanismo de pesquisa. Existem 5 principais características técnicas da computação em nuvem sendo eles recursos de computação em grande escala, alta escalabilidade e elasticidade, pool de recursos compartilhados (recursos virtuais e físicos), agendamento de recursos dinâmicos e uso geral".

Uma segunda definição do termo encontrada na literatura científica é: "A computação em nuvem descreve uma plataforma e um tipo de aplicativo. Uma plataforma de computação em nuvem provisiona, configura, reconfigura e desprovisiona servidores dinamicamente conforme necessário. Aplicativos em nuvem são aplicativos que são estendidos para serem acessíveis pela Internet. Esses aplicativos em nuvem usam grandes data centers e servidores poderosos que hospedam aplicativos da Web e serviços da Web." (BOSS et al., 2007)

Se apoiando nas definições de API apresentadas neste capítulo , é possível compreender a Nuvem como um conjunto de tecnologias, metodologias e ferramentas que fornecem um ambiente com um alto poder de escalabilidade, alta capacidade computacional, alta capacidade de armazenamento, redundância e facilidade no acesso ao recursos computacionais disponíveis, tudo isso em um modelo de cobrança onde o usuário paga apenas por aquilo que usar e em uma estrutura de recursos distribuídos em vários servidores alocados em diversos ISPs (*Internet Service Provider*) conectados através da internet e operados remotamente de qualquer lugar do mundo. É uma estrutura de recursos e serviços disponível a qualquer momento, de qualquer lugar.

2.2.2 Tipos de Nuvem

A Nuvem pode ser dividida em três tipos sendo eles pública, híbrida e privada. Segundo Goyal (2014), é possível definir esses três tipos da seguinte forma:

- **Nuvem pública**: a infraestrutura de Nuvem é disponibilizada ao público em geral ou a um grande grupo e é de propriedade de uma organização que vende serviços de Nuvem. Em Nuvens públicas os recursos são oferecidos como um serviço que não é cobrado;

- **Nuvem privada**: a infraestrutura de nuvem é operada exclusivamente para uma organização. Pode ser gerido pela organização ou por um terceiro e pode existir no local ou fora do local. A infraestrutura em nuvem é acessada apenas pelos membros da organização e/ou por terceiros concedidos. O objetivo não é oferecer serviços em nuvem para o público em geral, mas usá-lo dentro da organização;

- **Nuvem híbrida**: as Nuvens híbridas são mais complexas que os outros modelos de implantação, pois envolvem uma composição de duas ou mais Nuvens (privada, comunitária ou pública). Cada membro continua sendo uma entidade única, porém, está vinculado a outras por meio de tecnologia padronizada ou proprietária que permite a portabilidade de aplicativos e dados entre eles.

Existe ainda a *Community Cloud*) (Nuvem da comunidade). Segundo (ALZAIN et al., 2012) é possível definir esse novo tipo da seguinte forma:

- **Nuvem da comunidade (*Community Cloud*)**: a infraestrutura em Nuvem é provisionada para uso exclusivo por uma comunidade de consumidores de organizações que compartilham objetivos comuns (por exemplo, requisitos de segurança, política dentre outros). Pode ser propriedade, gerenciada e operada por uma ou mais organizações da comunidade, uma terceira parte, ou alguma combinação deles, e pode existir dentro ou fora das instalações das empresas.

2.2.3 Modelos de negócio

Quando se trata de Cloud existem, atualmente, três modelos de negócios que são utilizados para disponibilizar os recursos que ela abriga, são eles: IaaS – *infrastructure as a service* (infraestrutura como serviço), PaaS – *platform as a service* (plataforma como serviço) e SaaS – *software as a service* (software como serviço). Seguem suas definições segundo Mohammed, Zeebaree et al. (2021):

- **SAAS(*Software as a Service*)**: a oportunidade de usar aplicativos do provedor que operam em uma plataforma em Nuvem fornecida ao consumidor. Os aplicativos podem ser acessados por uma interface de aplicativo da web, como um navegador (por exemplo, um e-mail baseado na web) ou a interface do aplicativo, de dispositivos clientes separados;

- **IAAS(*Infrastructure as a Service*)**: a capacidade do cliente de fornecer computação, armazenamento, rede e outros serviços informáticos através dos quais o consumidor pode instalar e gerir máquinas virtuais, incluindo sistemas operacionais e aplicativos;

- **PAAS (*Plataform as a Service*)**: é a disponibilização sistemas operacionais e outros sistemas que são utilizados para implantar e desenvolver soluções na Nuvem. Quando um serviço de PAAS é contratado, normalmente, não existe a necessidade de se preocupar com a parte de infraestrutura associada à aquele recurso pois o provedor de serviços já cuidou desta parte. Ou seja, PAAS são sistemas e plataformas utilizados para suportar outros sistemas na nuvem (texto adaptado da obra Mohammed, Zeebaree et al. (2021)).

A Figura 2.2 ilustra a estrutura dos modelos de negócios na Nuvem.

A seguir, a Figura 2.3 ilustra uma arquitetura em nuvem e seus componentes.

2.3 O Conceito de cibersegurança

Segundo Craigen, Diakun-Thibault e Purse (2014) cibersegurança pode ser assim definida:

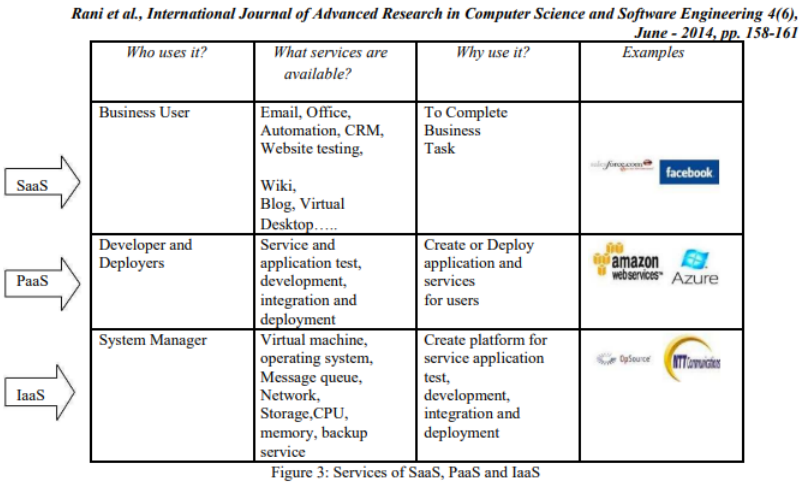

Figura 2.2: PaaS, IaaS e Saas (RANI; RANJAN, 2014)

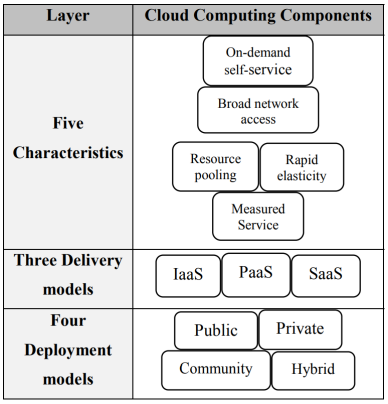

Figura 2.3: Arquitetura em Nuvem (MELL; GRANCE et al., 2011)

- a segurança cibernética consiste principalmente em métodos defensivos usados para detectar e impedir possíveis intrusos e outras ameaças a ativos digitais;
- a segurança cibernética envolve a proteção das redes de computadores e das informações que elas contêm contra penetração e danos ou interrupções maliciosas;
- a segurança cibernética envolve a redução do risco de ataques maliciosos a software, computadores e redes. Isso inclui ferramentas usadas para detectar invasões, parar vírus, bloquear acesso malicioso, impor autenticação, habilitar comunicações criptografadas e assim por diante.

Considerando as definições apresentadas no trabalho citado, é possível definir a Cibersegurança como um conjunto de metodologias e ações que visam proteger dados,

sistemas, programas, redes, dispositivos móveis e computadores contra ações que visam danificar, parcial ou totalmente, um ou mais ativos digitais ou físicos, públicos ou privados.

Segundo uma pesquisa realizada pela empresa responsável pelo antivírus McAfee, em 2020, o custo médio das atividade maliciosas que visam comprometer um ativo tecnológico, público e/ou privado, chegou a $945bi (Novecentos e quarenta e cinco bilhões de dólares) (SMITH, 2020).

Além das perdas monetárias existe ainda o fator humano. Muitas violações de sistemas acabam prejudicando a vida da população de alguma forma e, em casos mais sérios, podem levar à morte. Um exemplo disso é uma ação que foi realizada por um grupo hacker que alterava resultados de exames de tomografia adicionando manchas escuras nos resultados de exames de pacientes que estavam saudáveis, fazendo com que parecesse que o paciente possuía alguma enfermidade. Da mesma forma, o malware injetado pelo grupo fazia com que exames de pessoas que de fato possuíam algum tipo de enfermidade não acusassem nenhuma alteração, impedindo assim que a pessoa iniciasse o tratamento imediatamente.

Maiores detalhes a respeito desse lamentável ocorrido e sobre como um ataque desse tipo pode ser realizado podem ser encontrados em (MIRSKY et al., 2019).

A Figura 2.4, demonstra o cenário desse ataque.

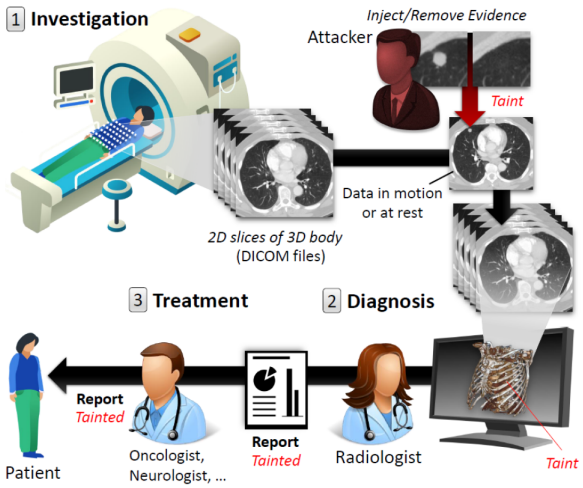

Figura 2.4: Execução de um ataque que modifica imagens de raio X (MIRSKY et al., 2019)

2.3.1 Trabalhos em andamento

Vários projetos foram desenvolvidos com o intuito de combater essas ameaças virtuais. Uma das iniciativas mais notáveis nesse âmbito é a OWASP (CERULLO, 2009). A OWASP tem como foco o teste de sistemas web, realizado em dezenas de milhares de APIs em todo o mundo. Com tamanha abrangência nos testes, a iniciativa lançou, em 2003, um ranking com as 10 brechas mais detectadas em sistemas web nos últimos 3 anos que antecedem a publicação dos resultados. Esse ranking é divulgado de 3 em 3 anos.

A Figura 2.5 mostra o resultado do último levantamento divulgado pela OWASP, onde é possível constatar o surgimento de 3 novas classes de vulnerabilidades.

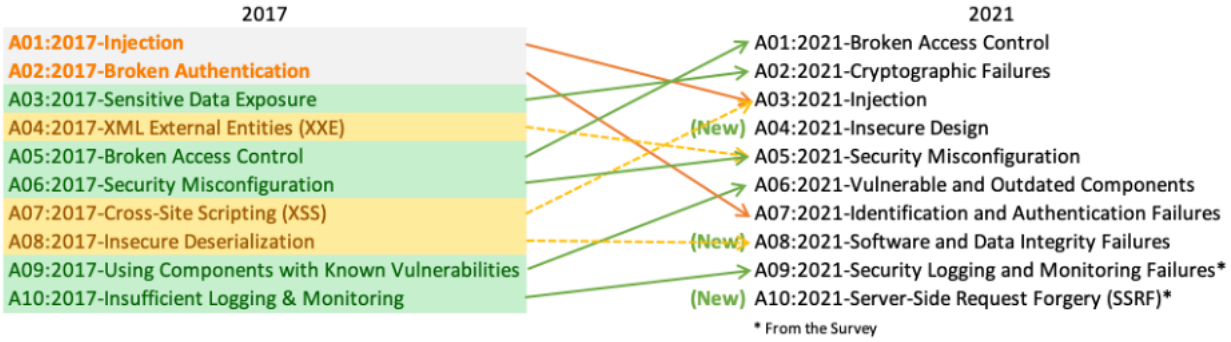

Figura 2.5: Top 10 classes de vulnerabilidades que mais apareceram em sistemas web no período de 2017 2021 (CERULLO, 2009)

Com o crescente uso de APIs a OWASP sentiu a necessidade de criar um top 10 das falhas mais comuns encontradas em APIs. As vulnerabilidades contidas no último top 10 (2017 2021) liberado pela OWASP são **Broken object level authorization**, **Broken authentication**, **Broken authentication**, **Excessive data exposure**, **Lack of resources and rate limiting**, **Broken function level authorization**, **Mass assignment**, **Security misconfiguration**, **Injection**, **Improper assets management** e **Insufficient logging and monitoring**.

Maiores detalhes acerca das vulnerabilidades encontradas nos levantamentos realizados pela owasp podem ser encontradas em seu site oficial (OWASP. . . ,).

A Nuvem é uma distribuição de recursos computacionais em constante evolução, mas apresenta muitos pontos que precisam ser melhorados, especialmente na segurança. Falhas em sistemas computacionais podem causar danos financeiros e até mesmo risco à

vida. Devido a isso, a segurança cibernética se tornou um tema cada vez mais relevante. No capítulo 3 serão apresentadas algumas pesquisas relacionadas e propostas de melhoria.

3 Trabalhos Relacionados

Neste capítulo serão apresentadas obras que tratam tanto de características do ambiente em Nuvem que interferem no funcionamento das APIs, bem como obras focadas especificamente na segurança delas. Ao fim, será feita uma comparação entre esses trabalhos.

3.1 Vulnerabilidades em Plataformas de Computação em Nuvem

A obra Ariffin, Ibrahim e Kasiran (2020) tem como objetivo abordar o tema de vulnerabilidades de APIs em plataforma de computação em Nuvem, exibindo definições que ajudam ao leitor entender conceitos importantes como por exemplo IAAS (*infrastructure as a service*), PAAS (*plataform as a service*) e SAAS (*software as a service*) entre outros.

A obra introduz o conceito de Nuvem mostrando ao leitor como esse ambiente tem sido cada vez mais utilizado, além disso explicita também que todo esse crescimento acabou atraindo também a atenção de pessoas mal intencionadas que têm por objetivo degradar serviços de terceiros expostos na internet.

Outras obras já foram feitas no sentido de discutir vulnerabilidades em APIs em Nuvem, o autor cita por exemplo Almutairy, Al-Shqeerat e Hamad (2019), Suryateja (2018) e (BASU et al., 2018), porém nenhuma delas trata especificamente do tipo de ataque de exaustão de API. O autor então foca neste tipo de ataque para demonstrar seu potencial destrutivo e como ele pode ser detectado.

Para conduzir os testes foi criado um ambiente controlado utilizando o software OpenStack (SEFRAOUI et al., 2012), uma plataforma de código aberto para gerenciamento de sistemas em Nuvem. Algumas outras opções de software de código aberto são citadas como por exemplo o Eucalyptus e o CloudStack, bem como softwares de código proprietário como o Citrix XenServer e o Vmware. Com o objetivo de emular uma configuração de provedor de Nuvem foi usado um servidor Dell OptiPlex 990 (3,40 GHz Intel

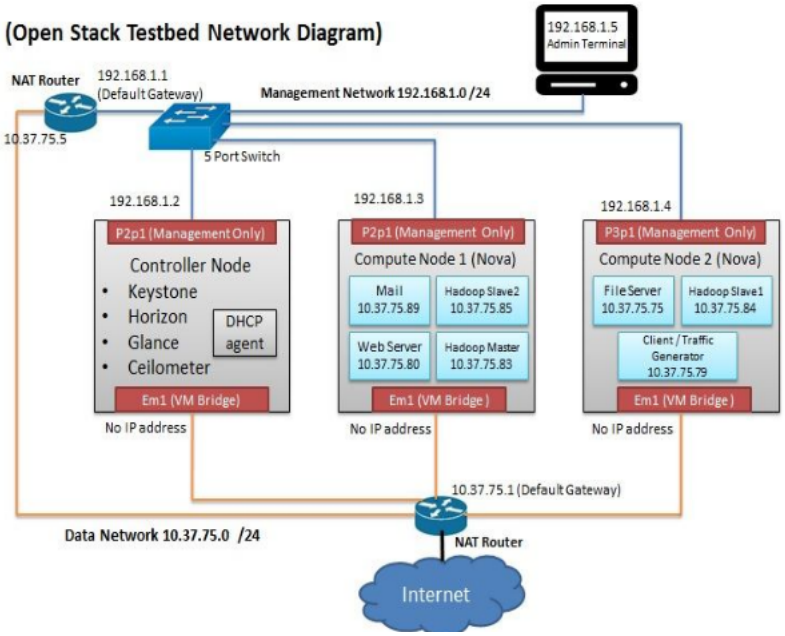

Figura 3.1: Arquitetura utilizada para testar o ataque de exaustão de API (ARIFFIN; IBRAHIM; KASIRAN, 2020)

i7-2600), 1 TB de disco rígido, 16 GB de memória, 2 interfaces de rede Gigabit-Ethernet. A figura 3.1 mostra a arquitetura utilizada.

Na obra são explorados dois tipos de ataques. O primeiro é um ataque de autenticação, onde o objetivo é conseguir credenciais de acesso que possam estar trafegando pela rede. O segundo é um ataque de exaustão de API, ou seja, um ataque no qual são feitas tantas requisições que o servidor não consegue processar a alta demanda de solicitações e começa a apresentar falhas.

Ao conduzir um ataque do tipo *man in the middle* (CONTI; DRAGONI; LESYK, 2016) utilizando o software *Wireshark*, o autor foi capaz de capturar payloads (CARGA. . . , 2022) de autenticação, conseguindo assim obter acesso a uma credencial que estava sendo transmitida pela rede conforme captura de tela exibida na Figura 3.2 na qual podemos ver a a senha **abcdef12345** que estava sendo transmitida pela rede sendo capturada pelo software **Wireshark**.

Ao conduzir um teste de exaustão de API a obra demonstra como esse tipo de ataque afeta um serviço e também como o modelo que é proposto na obra pode auxiliar na detecção dessa situação quando ainda em andamento baseado no algoritmo AD3 (MARTINS et al., 2012). A Figura 3.3 ilustra a estrutura desse modelo exibindos as fases

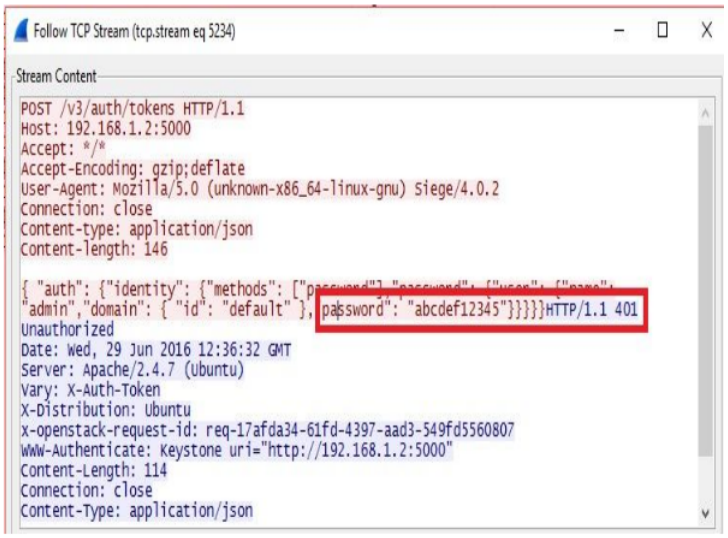

Figura 3.2: Credenciais interceptadas ao trafegar pela rede na conexão com o servidor de teste
(ARIFFIN; IBRAHIM; KASIRAN, 2020)

que ele possui e que o constituem sendo elas:

- **Receive System & Network Metric**: nesta fase são recebidos os dados de utilização de processador, memória, rede e disco do sistema que está sendo monitorado;
- **Feature Extraction/Selection**: nesta fase os dados recebidos na fase anterior são analisados e selecionados, como por exemplo qual é a utilização de processador em porcentagem? Ou ainda, Qual é a quantidade de memória utilizada ? Ou seja, nesta fase é dado um sentido para o dado;
- **Pre-Processing**: nesta fase os dados são normalizados;
- **Detection Algorithm**: nesta fase é executado o algoritmo de detecção, sendo ele o responsável por identificar se existe um ataque em curso ou não com base nos dados normalizados recebidos da fase anterior. Esses dado poem ser direcionados para alguma saída de visualização para um agente externo possa tomar alguma decisão, essa etapa é denominada no modelo por Analysis & Visualization;
- **Identification**: nesta fase a ameaça, caso detectada, será então identificada, ou seja, será categorizada, ou seja, será feita a identificação paar saber qual é o agente ofensor do sistema.

- **Decision**: por fim se segue a fase de decisão, onde será decidido qual decisão será tomada acerca do que foi detectado pelo algoritmo.

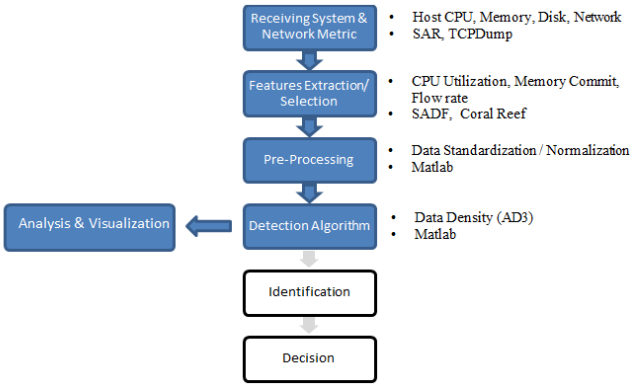

Figura 3.3: Modelo proposto para detectar anomalias em recursos computacionais em Nuvem
(ARIFFIN; IBRAHIM; KASIRAN, 2020)

O trabalho (KADAM, 2011) traz uma visão transparente sobre o ambiente da Nuvem, focando especificamente em questões relacionadas à segurança. Para tal, o autor se apoia em várias outras obras já publicadas com intuito de extrair definições que auxiliam o leitor a compreender melhor em que consistem os problemas exibidos.

A obra primeiramente introduz o conceito de Nuvem, citando a seguinte definição "Cloud Computing é um modelo criado para permitir acesso de rede onipresente, conveniente e sob demanda a um pool compartilhado de recursos de computação configuráveis (rede, servidor, armazenamento, aplicativo e serviços) que podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou provedor de serviços interação." (JANSEN; GRANCE,).

São apresentadas definições para os modelos de serviço mais comuns praticados pelos provedores sendo eles IAAS(*infrastructure as a service*), PAAS(*plataform as a service*) e SAAS(*software as a service*).

É mostrado que esses problemas de segurança impactam na forma como esses modelos de serviços são executados pois em uma arquitetura em Nuvem que possua elementos desde o nível de aplicação, ou seja, a parte da arquitetura que faz interface com o usuário até a parte que roda na nuvem, o CSP fica limitado no que diz respeito à medidas de segurança conforme a arquitetura se aproxima do usuário, dada a dificuldade de

monitorar e controlar os dispositivos finais, ou seja os dispositivos dos consumidores. Isso está sintetizado na Figura 3.4.

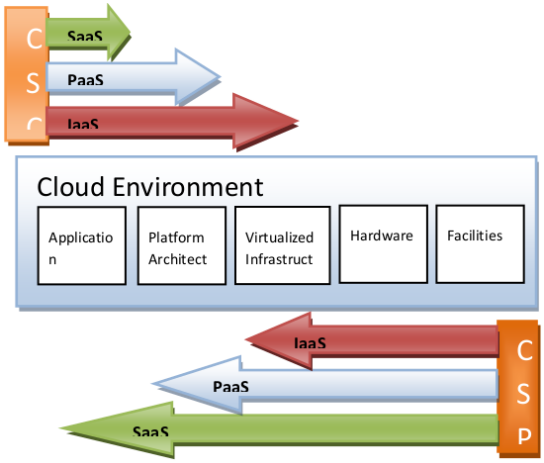

Figura 3.4: Escopo e controle entre o modelo de serviço em Nuvem (KADAM, 2011)

O autor destaca que a computação em Nuvem está ainda em sua fase inicial e por isso é extremamente importante coletar dados a partir da experiência dos usuários para detectar e mitigar riscos relacionados à segurança, sendo esse o fator mais preocupante.

São citados pontos importantes que podem se tornar uma fonte de problemas quando não são gerenciados corretamente sendo eles SLA (*Service Level agreement*)/QOS (*Quality of Service*), IDaaS (*Id as a service*), Governança, Conformidade, Proteção de dados, Arquitetura, Segurança de Email, Segurança de rede, Segurança da Web, Detecção de Intrusão, Backup e Recuperação

Para cada um dos pontos levantandos acima o autor traz situações que podem levar a uma diminuição do nível de segurança.

A obra (PETRILLO et al., 2016), de título "Are rest APIs for cloud computing well-designed?", traz um compilado de 73 regras de boas práticas para se adotar no ambiente da nuvem, principalmente para CSPs que adotam o suporte a APIs para seu funcionamento. O autor identifica que, apesar do grande crescimento, o ambiente da Nuvem ainda possui muitos pontos a melhorar, como exemplo é citado o GCP(Google Cloud Platform) que possui suas próprias APIs em código fechado, por outro lado existe por exemplo o OpenStack (SEFRAOUI et al., 2012) que possui sua API em código aberto, ou seja, existem vários padrões com os quais é necessário lidar. A obra cita as contribuições

do trabalho (MASSE, 2011) que propôs 65 boas práticas para se adotar ao arquitetar uma API REST. Outras obras citadas pelo autor que trazem boas práticas são (RODRÍGUEZ et al., 2016), (PALMA et al., 2015b) e (PALMA et al., 2015a).

O autor estabelece quatro questões que guiam os estudos realizados no trabalho, sendo elas :

- **Q1**: Quais são os principais serviços fornecidos pelas APIs REST da nuvem?
- **Q2**: Quantas práticas recomendadas são seguidas pelas APIs REST da nuvem?
- **Q3**: Quais práticas recomendadas são adotadas por todas as APIs?
- **Q4**: Quais práticas recomendadas não são adotadas por nenhuma das APIs?

As platformas escolhidas pelo autor da obra foram o Google Cloud Patform (BISONG, 2019), OpenStack (SEFRAOUI et al., 2012) e OCCI(Open Cloud Computing Interface) (METSCH; EDMONDS; PARÁK, 2010).

A metodologia adotada pela obra foi a conferência manual. O autor testa manualmente cada uma das três plataformas citadas acima, para cada uma das 73 boas práticas compiladas. O padrão OCCI, inclusive, foi validado também por dois colaboradores que auxiliam no desenvolvimento do padrão. Os resultados foram compilados e mapeados em várias tabelas comparativas exibidas ao longo do artigo.

A obra (JING; JIAN-JUN, 2010), de título "brief survey on the security model of cloud computing", traz uma análise de uma série de problemas de segurança que podem ocorrer em ambiente de computação em Nuvem. Além disso, a obra propõe também um modelo de segurança para ajudar a mitigar os problemas apontados ao longo do estudo, além disso, traz exemplos de arquiteturas em Nuvem, como mostrado na Figura 3.5

Na obra em questão foram levantados alguns problemas relacionados à segurança da nuvem, incluindo a dificuldade em definir limites claros para a Nuvem o que pode dificultar a proteção do usuário final bem como a possibilidade de dados pessoais dos usuários, serviços e ativos estarem em risco em caso de falha nos servidores do CSP (Cloud Service Provider), o que pode ser agravado pelo fato de que os usuários utilizam muitos serviços diferentes, o que torna difícil a sua classificação e a identificação de usuários mal intencionados.

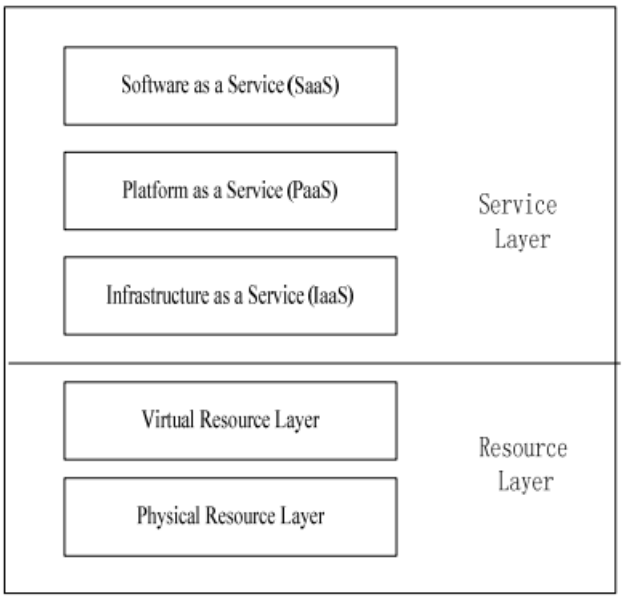

Figura 3.5: Exemplo de arquitetura em Nuvem disponível na obra (JING; JIAN-JUN, 2010)

Para mitigar esses riscos, uma proposta de política de segurança apresentada na obra sugere a divisão de domínios de segurança da aplicação, a garantia da segurança da conexão do usuário com os serviços internos do CSP através de tecnologias como SSL, VPN e PPTP, a proteção dos dados dos usuários, o monitoramento das ações de sistemas de terceiros e o monitoramento das ações dos usuários para identificar possíveis comportamentos mal intencionados

Após a definição da polítia de segurança acima, o autor introduz o conceito de SACS (Serviço de Controle de Acesso de Segurança). Esse modelo inclui autorização de acesso e API de segurança e segurança de conexão em Nuvem.

A Figura 3.10 ilustra esse modelo o modelo de segurança proposto pelo autor.

O trabalho (SURYATEJA, 2018) tem por objetivo trazer várias ameaças que permeiam o ambiente da Nuvem e que por consequência podem afetar diretamente o uso de APIs que porventura venham a estar hospedadas.

Além da definição de Nuvem o autor traz em sua obra características importantes que devem ser levadas em consideração ao se projetar uma API que irá rodar na Nuvem, sendo elas autoatendimento sob demanda, amplo acesso à rede, agrupamento de recursos e serviço medido.Além disso, a obra também nos traz definições para os termos de IAAS, PAAS e SAAS.

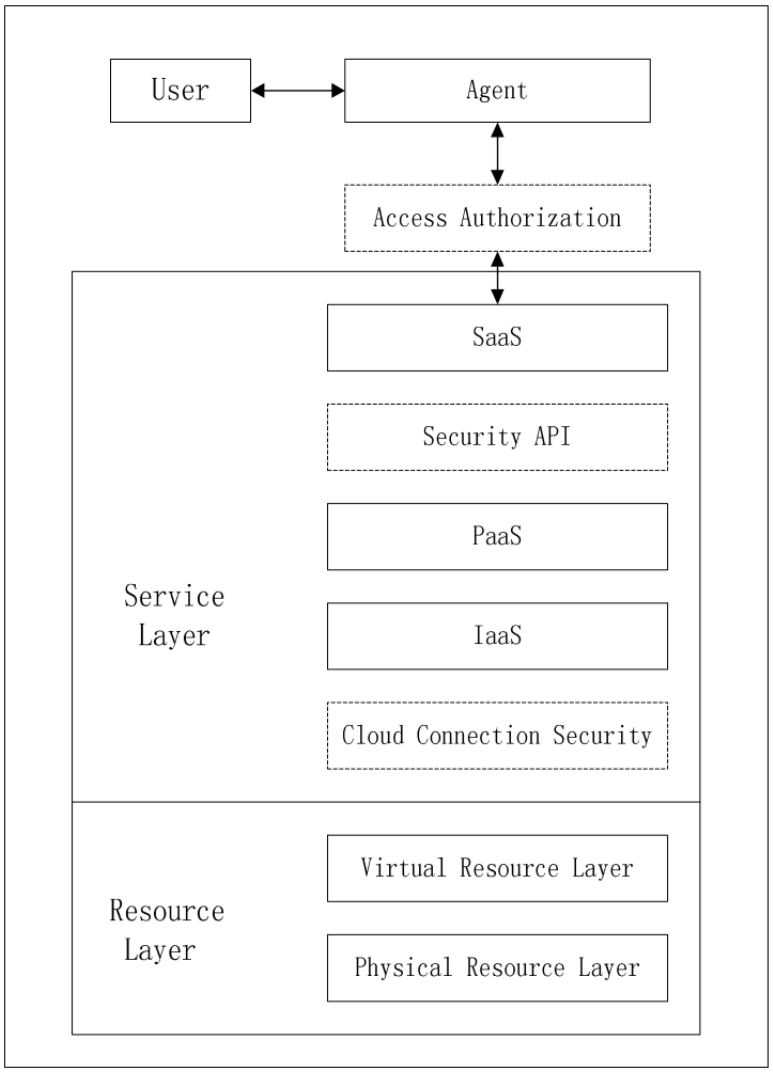

Figura 3.6: Modelo de segurança proposto pela obra (JING; JIAN-JUN, 2010)

É apresentado, ainda, um estudo realizado por um grande provedor de SAAS, a RightScale, que mostra que a segurança é um dos principais desafios encontrados pelas empresas que já atuam ou que desejam atuar no ambiente da Nuvem. A figura 3.12 mostra este estudo.

Um ponto interessante de se notar nesta obra é que ela traz ameaças recentes como por exemplo Specter e Meltdown (KHARRAZ et al., 2015) e os Ransomwares (AHMET; GÜNGÖR, 2019). Além desses, também são exibidas outras ameaças, sendo elas:

- **Violações de Dados**: isso ocorre quando dados sensíveis são vazados;
- **Perda de Dados**: ocorre quando dados importantes são perdidos;

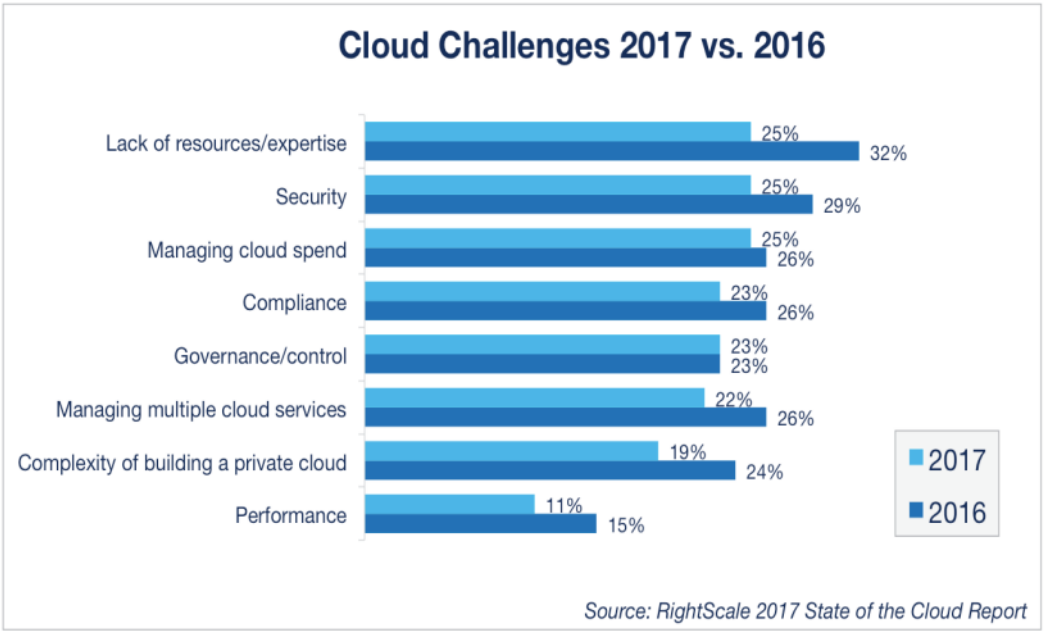

Figura 3.7: Estudo realizado pela RightScale mostra os principais desafios para se operar no ambiente da Nuvem
(SURYATEJA, 2018)

- **Internos Maliciosos**: essa situação é especialmente perigosa pois um funcionário mal intencionado que tenha acesso a sistemas críticos pode modificar, excluir ou adicionar arquivos nos servidores do CSP, comprometendo todo o funcionamento das APIs e outros sistemas;

- **Negação de Serviço (DOS ou DDOS)**: ataques de negação de serviço podem derrubar uma API inteira e até mesmo outros serviços que estejam sendo executados na mesma máquina que a API alvo pois esgotam os recursos da máquina impedindo que outros serviços sejam executados. Exemplos de DOS e DDOS são vistos nas figuras 3.8 e 3.9 respectivamente;

- **Ransomware**: o ransomware é um software malicioso que criptografa os arquivos do computador infectado e então exige um resgate, geralmente pago em bitcoins (WAMBA et al., 2020), para que os arquivos sejam descriptografados. Mesmo pagando o resgate, não há garantia de recuperação dos arquivos;

- **Spectre and Meltdown**: essas duas ameaças formam um conjunto de vulnerabilidades descoberta em vários processadores das marcas Intel e AMD. Ainda estão

presentes em muitos sistemas desatualizados e que permitem a um atacante executar códigos remotamente em uma máquina alvo;

- **Sequestro de Conta**: ocorre quando um atacante consegue obter as credenciais ou informações de seçaõ de uma conta de um usuário legítimo, comprometendo assim todo o sistema/servidor;

- **Erro Humano**: erros humanos constituem uma grande ameaça a qualquer sistema pois dependendo de quão grave for o erro existe o risco de que todo o servidor e/ou API/sistema seja comprometido, seja imediatamente ou a longo prazo.

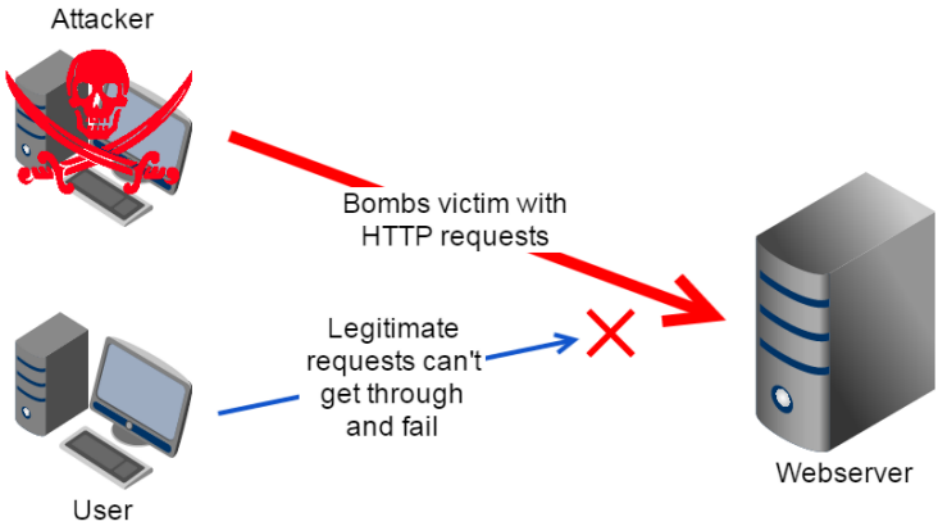

Figura 3.8: Exemplo de como ocorre um ataque do tipo DOS (AHMET; GÜNGÖR, 2019)

A obra (SUBRAMANIAN; JEYARAJ, 2018) traz um compilado de desafios relativos à segurança no ambiente da computação em Nuvem bem como uma série de medidas que podem ser tomadas para mitigar os riscos apresentados.

A obra define uma arquitetura em Nuvem como sendo composta por duas partes, *front-end* e *back-end*, sendo o *front-end* a parte que faz interface com o usuário e o *back-end* a parte onde o usuário não tem acesso, ou seja, a parte do sistema ou API que é executada no CSP. O autor apresenta uma definição simplificada de uma arquitetura em Nuvem que pode ser vista na Figura 3.10.

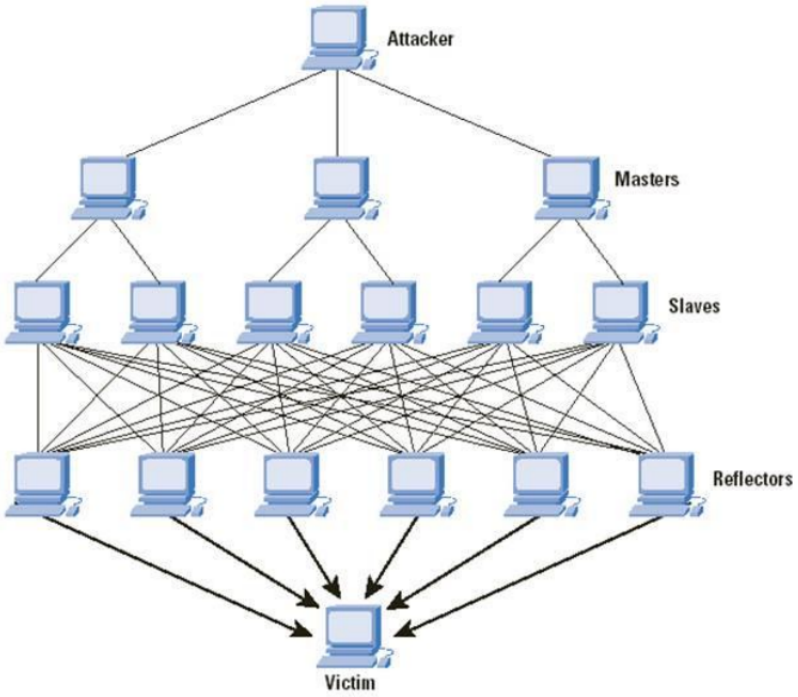

Figura 3.9: Exemplo de como ocorre um ataque do tipo DDOS
(AHMET; GÜNGÖR, 2019)

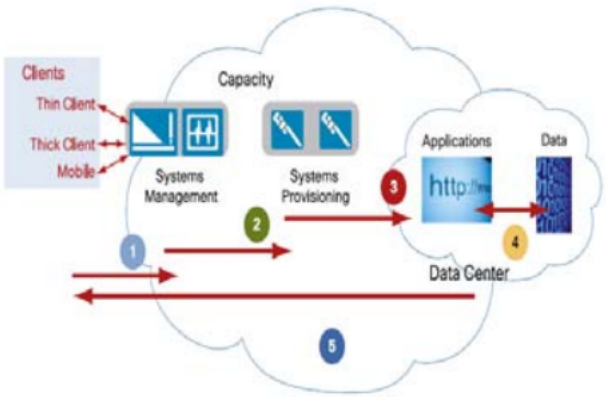

Figura 3.10: Exemplo de arquitetura simples em Nuvem
(SUBRAMANIAN; JEYARAJ, 2018)

A obra traz algumas aplicações do ambiente de computação em Nuvem por exemplo a facilidade de acesso dos clientes às suas aplicações, a opção das organizações de poderem alugar máquinas sem a necessidade de tê-las fisicamente, redução do de hardware entre outras.

Apesar de seu grande crescimento o ambiente da Nuvem ainda precisa melhorar alguns pontos. Aqui a obra traz um compilado de situações que devem ser levadas em conta ao se projetar uma API ou serviço para este ambiente:

- **Perder o Controle Sobre os Dados**: existe o risco dos dados serem roubados e por isso a aplicação precisa estar preparada para conter esse risco;
- **Integridade dos Dados**: Muitos ataques visam corromper os dados, comprometendo assim sua integridade;
- **Problema de Incompatibilidade**: as vezes um serviço x fornecido por um CSP é incompatível com o ambiente y de outro CSP;
- **Adições Constantes de Recursos**: a todo instante aplicações em Nuvem recebem recursos, os usuários precisam se manter atualizados;
- **Falha na Segurança do Provedor**: um CSP pode ser alvo de ataque e isso pode levar ao comprometimento dos serviços que ele hospeda;
- **Ataque Distribuído de Negação de Serviço (DDOS)**: um ataque distribuído pode comprometer todos os serviços que rodam em determinado CSP e até mesmo outros CSPs que dependam dele;
- **Ataque de Homem no Meio**: ataques *man in the middle* são muito perigosos por serem capazes de levar a vazamentos de credenciais de acesso e outras informações importantes;
- **Localização dos Dados**: dependendo do tipo de aplicação o local onde o dados residem pode se tornar um problema caso esse local não apresente uma segurança adequada.

Para ajudar a mitigar possíveis riscos existentes no ambiente da Nuvem, existem alguns padrões de segurança que podem ser aplicados, o autor ciat alguns, sendo eles Linguagem de Marcação de Asserção de Segurança (SAML), Autenticação Aberta (OAuth), OpenID e SSL/TLS.

Ao analisar as obras apresentadas é possível verificar que o campo da computação em Nuvem apresentou um avanço muito grande desde sua criação. Porém este é um ambiente que ainda esconde muitos riscos e desafios, do ponto de vista de segurança em APIs, esses riscos são especialmente preocupantes pois quando explorados podem causar sérios danos às empresas, sistemas ou até mesmo à vidas humanas. A seguir é apresentada uma tabela comparativa destacando pontos positivos, negativos, ferramentas utilizadas e metodologia empregada entre as obras abordadas.

Obra	Metodologia	Ferramentas	Pontos Positivos	Pontos Negativos
Security Challenges in Cloud Computing	Levantamento manual	Não informadas	São apresentados modelos de segurança e falhas críticas de sistemas	Exemplos de aplicações dos padrões de segurança propostos ausentes
API Vulnerabilities In Cloud Computing Platform: Attack And Detection	Teste manual	Software OpenStack, wireshark, servidor físico Dell optiplex 990	Realização de testes em cenário real com situações reais	Não definição técnica de alguns termos, por exemplo man-in-the-middle
Security Issues in Cloud Computing A Transparent View	Levantamento manual de pontos de falha	Não informadas	Traz pontos de falha que normalmente não tratados em outro artigos,	Não exibir cenário de exploração das falhas relatadas
Are REST APIs for Cloud Computing Well-Designed? An Exploratory Study	Conferência manual	Não informadas	Traz um grande compilado de boas práticas utilizadas ao se criar uma APIN	Não traz exemplos concretos problemas que surgem ao não seguir tais práticas
A brief survey on the security model of cloud computing	Levantamento e enumeração manual	Hadoop	Modelo de segurança proposto é extensível	Tabela de resultados confusa, falta de detalhes a respeito do ambiente de testes utilizado
Threats and vulnerabilities of cloud computing: A review	Levantamento manual	Não informadas	Exposição de uma grande variedade de falhas possíveis, traz falhas recentes como Specter e Meltdown	Falta de um exemplo concreto de como essas falhas poderiam ser prejudiciais

Tabela 3.1: Síntese das obras analisadas no capítulo.

A Tabela 3.1 oferece uma visão geral das obras analisadas, destacando metodologias, ferramentas utilizadas, pontos positivos e negativos de cada estudo sobre segurança na computação em nuvem. Ela revela a diversidade de abordagens, desde levantamentos manuais até testes em cenários reais, com ferramentas variadas. Os pontos positivos ressaltam a identificação de modelos de segurança, boas práticas e exposição de falhas críticas, mas algumas obras carecem de exemplos concretos ou aplicação prática das soluções propostas, enfraquecendo o impacto da pesquisa. Há uma falta de detalhes em alguns estudos, como a não especificação de ferramentas ou a confusão na apresentação dos resultados, o que pode prejudicar a compreensão e aplicabilidade dos achados. Essa diversidade e lacunas destacam a necessidade contínua de estudos que não apenas identifiquem falhas, mas também ofereçam soluções práticas e demonstrações claras de suas implicações na segurança das APIs

É possível fazer um paralelo entre as obras analisadas e esta, considerando os aspectos destacados na tabela:

- **Metodologia**: testes manuais e automatizados utilizando ferramentas de pentest bem como padrões de testes disponíveis no OWASP WSTG;
- **Ferramentas**: OWASP ZAP, Subfinder, Wapiti, Subjack, Nikto, Arachni, Burp Suite. Também foram utilizadas ferramentas não específicas para pentest mas que podem ser utilizadas para essa finalidade, sendo elas o google dorks, Navegador da web Chrome, Postman, Android ADB, Genymotion;
- **Pontos Positivos**: como pontos positivos desta obra quando comparada às outras que foram mostradas no capítulo é possível elencar o grande número de APIs analisadas e a indicação de um conjunto de ferramentas e ações que podem ajudar a mitigar os defeitos encontrados;
- **Pontos Negativos**: como ponto negativo é possível destacare o fato de que não foi explorado no texto como essas ferramentas podem ser utilizadas para efetivamente corrigir os defeitos encontrados nas APIs.

4 Experimentos e Resultados

4.1 Experimentos

Um estudo exploratório abordou a análise de 200 APIs, tanto móveis quanto web, utilizando um conjunto diversificado de ferramentas, conforme descrito na tabela. Além disso, o estudo incorporou o framework de testes da OWASP WSTG (Web Security Testing Guide). As ferramentas selecionadas incluem Wapiti, Nikto, Owasp Zap, Arachni, Subfinder, Subjack, Burp Suite, juntamente com o referido framework.

A escolha dessas ferramentas foi fundamentada em critérios específicos. A preferência por ferramentas de código aberto foi motivada pelo desenvolvimento ativo e atualizações frequentes que caracterizam esse tipo de software. A natureza aberta permite acesso ao código-fonte, possibilitando consultas e modificações conforme necessário.

Outro critério de seleção foi o nível de detalhamento e formato dos relatórios fornecidos por essas ferramentas. Priorizou-se aquelas capazes de oferecer relatórios detalhados, permitindo uma compreensão aprofundada das vulnerabilidades identificadas. Além disso, a capacidade de integração com outras ferramentas foi considerada crucial para um processo de análise abrangente.

Essas ferramentas, selecionadas criteriosamente, são representativas de diferentes aspectos de segurança, abordando desde scanners de vulnerabilidades a ferramentas de descoberta de subdomínios. Esse conjunto diversificado proporcionou uma visão ampla e aprofundada das questões de segurança enfrentadas pelas APIs, permitindo uma análise abrangente e detalhada das possíveis vulnerabilidades.

Após a escolha da metodologia de teste que seria utilizada, foi definido um plano de testes para guiar a pesquisa a ser realizada. Este plano de testes desempenhou um papel fundamental, pois nele foram definidas importantes diretrizes que orientaram todos os testes, como por exemplo a coleta e análise dos resultados, escopo, localização e conjunto de ferramentas. Isso ajudou a garantir um padrão que pudesse ser aplicado a todas as APIs analisadas, permitindo assim que todas fossem avaliadas seguindo as mesmas diretrizes,

além de definir de forma clara os limites dos testes a serem realizados.

Para selecionar os sistemas que seriam testados foram adotados os seguintes critérios:

- **APIs web**: no que diz respeito à web, o único critério utilizado foi o fato de estar ou não acessível publicamente. Quanto mais fácil for localizar determinado sistema, melhor;
- **APIs mobile**: para escolher os aplicativos mobile que seriam alvos de testes, foi utilizado como critério o fato de um determinado aplicativo ter ou não uma nota baixa na loja de aplicativos escolhida, que, no caso, foi a Play Store, bem como o número de reclamações acerca do aplicativo. Quanto mais reclamações um aplicativo possui, especialmente reclamações acerca do funcionamento do aplicativo, mais falhas ele tende a ter.

A aplicação das diretrizes de teste do WSTG seguiu um procedimento dinâmico e abrangente. Sempre que possível, as ferramentas escolhidas foram empregadas para realizar os testes de acordo com as diretrizes delineadas. Quando uma tarefa não era compatível com as ferramentas selecionadas, os testes eram conduzidos manualmente, valendo-se de recursos como Postman e navegadores web. Essa abordagem híbrida permitiu uma cobertura extensa, tanto automatizada quanto manual, garantindo a exploração completa das vulnerabilidades potenciais das APIs analisadas. Conforme o estudo era realizado as evidências eram coletadas para posterior análise.

4.2 Resultados

Os resultados encontrados nesta pesquisa exploratória foram consolidados em uma tabela contendo os endpoints onde ocorreram as falhas (com os dados devidamente anonimizados), qual metodologia do framework de testes WSTG encontrou aquela falha, quais CWEs estão associados a essa falha e qual conjunto de ferramentas foi utilizado no teste, conforme a **Tabela 6.1** presente no **Apêndice 1** localizado ao final deste estudo.

A seguir é exibida uma síntese dos resultados do estudo exploratório realizado:

Total de APIs Analisadas	Apis com Falhas	Taxa Média de CWE por API	CWEs de Maior Frequência	CWEs de Menor Frequência
200	95 (48%)	210/200=1.05	CWE-200	CWE-521

Tabela 4.1: Síntese dos Resultados

CWE	Ocorrências
CWE-22: Path Traversal	3
CWE-16: Configuration	20
CWE-703: Improper Check or Handling of Exceptional Conditions	6
CWE-254: Security Features	6
CWE-200: Exposure of Sensitive Information to an Unauthorized Actor	39
CWE-201: Information Exposure Through Sent Data	26
CWE-311: Missing Encryption of Sensitive Data	4
CWE-2000: Information Exposure	11
CWE-215: Information Exposure Through Debug Information	10
CWE-256: Unprotected Storage of Credentials	24
CWE-522: Insufficiently Protected Credentials	22
CWE-330: Use of Insufficiently Random Values	8
CWE-89: SQL Injection	4
CWE-287: Improper Authentication	4
CWE-285: Improper Authorization	3
CWE-542: Authentication Bypass by Capture-replay	6
CWE-320: Key Management Errors	6
CWE-532: Inclusion of Sensitive Information in Log Files	1
CWE-601: URL Redirection to Untrusted Site	6
CWE-521: Weak Password Requirements	1
Total	210

Tabela 4.2: Frequência de Ocorrências de CWEs. Para saber a qual falha um código cwe faz referência consultar o site oficial: https://cwe.mitre.org

Ao analisar os resultados é possível verificar pontos importantes como por exemplo:

- a grande quantidade de vazamentos de dados, como credenciais de desenvolvimento, dados pessoais de clientes, entre outros.
- o fato de que quase 50% das APIs apresentaram pelo menos uma falha.
- Uma grande parte das falhas se origina, muito provavelmente, em estágios anteriores ao desenvolvimento, como o estágio de planejamento e definição de comportamento da API.
- o quão fácil é achar essas falhas. Por exemplo, falhas de configuração de permissões

em cloud storages, como buckets AWS/Google Cloud/Azure, podem ser localizadas utilizando apenas o navegador. Isso é muito preocupante, pois quanto mais fácil é achar essas falhas, mais usuários mal intencionados podem tentar explorar essas brechas.

- o quão facilmente essas brechas poderiam ser corrigidas, indicando que não foram realizados (ou foram realizados de forma errada) testes de segurança nessas APIs/-configurações.

- como a facilidade de acesso a recursos na nuvem contribui para o aumento de APIs defeituosas. A facilidade de um usuário comum conseguir um domínio tem se mostrado algo que pode trazer alguns problemas. Um deles é permitir que falhas como subdomain takeover e phishing se tornem cada vez mais frequentes.

Muitas das falhas encontradas, na grande maioria dos casos, poderiam ser facilmente corrigidas. Isso mostra que o processo de desenvolvimento utilizado para desenvolvê-las foi/é defeituoso, carecendo de estágios de testes de segurança, o que acaba por comprometer a segurança dos usuários. Isso é um fato preocupante, dado que quanto mais brechas um sistema possui, maiores são as chances de que alguma delas venha a ser explorada.

Um ponto importante a ser destacado é que, durante esses testes, foram realizados apenas testes não invasivos, ou seja, testes que tendem a localizar as brechas que estão mais facilmente detectáveis. Se esses mesmos sistemas fossem submetidos a testes de intrusão ainda mais completos, existem grandes chances de que ainda mais falhas fossem detectadas.

Com isso esse estudo evidencia algumas vulnerabilidades que podem surgir em APIs, principalmente naquelas que são desenvolvidas para serem executadas na Nuvem, demonstrando como esse ambiente possui um grande potencial, porém ainda é necessário um maior esforço para se compreender melhor o que é e como esse novo paradigma funciona.

Além disso, é possível elencar algumas ferramentas que poderiam ser utilizadas para detectar as falhas aelencadas nesta obra. São elas:

Ferramenta	Wapiti	Nikto	OWASP ZAP	Arachni	Subfinder	Subjack	Burp Suite
Tipo de Ferramenta	Scanner de Segurança Web	Scanner de Vulnerabilidades Web	Scanner/Proxy de Segurança Web	Scanner de Segurança Web	Encontrar Sub-domínios	Encontrar Sub-domínios	Scanner de Vulnerabilidades
Licença	GPL v2	GPL	Apache 2.0	Apache 2.0	MIT	MIT	Grátis e Paga
Linguagem de Programação	Python	Perl	Java	Ruby	Go	Go	Java
Plataformas Suportadas	Linux, Windows, macOS	Linux, Windows, macOS	Linux, Windows, macOS	Linux, Windows, macOS	Windows, Linux e macOS	Windows, Linux e macOS	Windows, Linux e macOS
Interface Gráfica (GUI)	Não	Não	Sim	Não	Não	Não	Sim
Capacidade de Scripting	Não	Não	Sim	Sim	Não	Não	Sim (Versão Paga)
Modo de Operação	Console	Console	GUI / Texto / Console / Automação	Console	Console	Console	GUI / Texto / Console / Automação
Varredura Automática	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Integração com Outras Ferramentas	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Relatórios Detalhados	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Comunidade Ativa	Moderada	Moderada	Ativa	Ativa	Ativa	Moderada / baixa	Ativa
Atualizações Regulares	Não	Não	Sim	Sim	Sim	Não	Sim

Tabela 4.3: Conjunto de ferramentas para detecção de brechas encontradas

Vale ressaltar que durante o estudo foram utilizadas outras ferramentas que não são específicas para pentest, porém, também podem auxiliar na localização destas falhas, são elas:

- **Postman**: o postman é uma plataforma de colaboração para o desenvolvimento de APIs, permitindo testar, documentar e compartilhar endpoints de maneira simplificada e eficiente, agilizando o processo de desenvolvimento e integração de serviços web;

- **Google dork**: google dorks são consultas específicas no Google, usando operadores avançados para encontrar dados sensíveis em sites ou servidores, expondo informações confidenciais que não foram protegidas adequadamente, revelando vulnerabilidades;

- **Navegador(web browser)**: um navegador da web, ou web browser, é um aplicativo que permite aos usuários acessar e visualizar informações na internet. Ele interpreta e exibe conteúdo web, como páginas, vídeos e imagens, por meio da interação com servidores web permitindo analisar e modificar payloads enviados e recebidos através das APIs dos dispositivos. Neste estudo o navegador utilizado foi o google chrome;

- **Android adb**: o adb é uma ferramenta de linha de comando para interagir com dispositivos Android, possibilitando depuração, instalação de apps e controle remoto avançado. Essencial para desenvolvedores, ele viabiliza testes e operações de gerenciamento de dispositivos via terminal;

- **Genymotion**: o genymotion é uma plataforma de emulação de dispositivos android, fornecendo uma ampla gama de dispositivos virtuais para testar aplicativos. Com alta performance e integração simplificada, é uma ferramenta eficaz para acelerar o desenvolvimento de aplicativos móveis

Mais informações acerca das ferramentas citadas acima podem ser encontradas nos seguintes endereços:

- **Wapiti**: wapiti-scanner.github.io;

- **Nikto**: github.com/sullo/nikto;
- **Owasp zap** : www.zaproxy.org;
- **Arachni**: github.com/Arachni/arachni;
- **Subfinder**: github.com/projectdiscovery/subfinder;
- **Subjack**: www.kali.org/tools/subjack;
- **Burp suite**: portswigger.net/burp;
- **Postman**: www.postman.com;
- **Google dork**: www.exploit-db.com/google-hacking-database;
- **Navegador(web browser)**: www.google.com/intl/pt-BR/chrome;
- **Android adb**: developer.android.com/tools/adb?hl=pt-br
- **Genymotion**: www.genymotion.com

4.3 Como Mitigar

Ainda se baseando nas falhas observadas, é possível elencar também algumas medidas que podem ser tomadas para localizar essas (e outras) falhas, permitindo, assim, aumentar o nível de segurança dos sistemas expostos na web. São elas:

- **Testes de Intrusão (penetration testing)**: realizar testes de penetração pode ajudar a identificar vulnerabilidades como Path Traversal, SQL Injection, Improper Authentication, Improper Authorization e outras falhas de segurança, explorando diretamente a aplicação para encontrar pontos fracos;
- **Análise estática do código**: utilizar ferramentas de análise estática de código para identificar possíveis vulnerabilidades e falhas de segurança no código-fonte, como configurações incorretas, manipulação inadequada de exceções, entre outros pontos;

- **Análise dinâmica de segurança (dynamic security analysis)**: executar ferramentas de análise dinâmica para examinar o comportamento da aplicação em tempo de execução, procurando por falhas como Information Exposure, Unprotected Storage of Credentials, entre outras;

- **Revisão de requisitos de segurança**: rever os requisitos de segurança estabelecidos para a aplicação, garantindo que os princípios de segurança, como criptografia, autenticação forte, controle de acesso, estejam adequadamente implementados. Muitos sistemas sequer possuem uma definição clara dos requisitos de segurança que devem ser seguidos. Muitas vezes esse fator é completamente desconsiderado em favor da velocidade de produção e disponibilização de determinada API/recurso;

- **Auditoria de logs e monitoramento**: revisar registros de eventos (logs) da aplicação para identificar atividades suspeitas, possíveis falhas de segurança ou tentativas de ataques;

- **Revisão de configuração de segurança**: avaliar e garantir que as configurações de segurança, como criptografia de dados, gerenciamento de chaves, configurações de firewall, estejam corretas e eficazes;

- **Testes de interface**: considerar testes que explorem a usabilidade da aplicação e a interface do usuário em busca de possíveis brechas de segurança pode ser uma boa medida para promover o aumento da segurança do software;

- **Testes de autenticação e autorização**: realizar testes específicos para verificar a robustez dos mecanismos de autenticação e autorização, tentando contornar ou burlar esses sistemas para identificar possíveis brechas ou falhas de acesso indevido.

- **Uso de ferramentas baseadas em IA**: o uso de ferramentas baseadas em IA pode ajudar a identificar falhas que as ferramentas baseadas em análise estática não são capazes de identificar, como por exemplo fluxos mal desenhados.

Proteger APIs impõe novos desafios e esses desafios exigem novas abordagens para que as devidas falhas sejam encontradas e corrigidas. Falhas que não estão relacionadas

a regras de negócio, por exemplo, têm sido cada vez mais exploradas por usuários mal-intencionados, visando comprometer, de alguma forma, a integridade do serviço alvo. As ações maliciosas agora se voltam à lógica empregada na construção do recurso e não apenas em suas características tecnológicas. Hoje, hackear a lógica por trás de algum serviço ou recurso significa hackear a lógica da pessoa que construiu aquele código; significa encontrar brechas na forma de pensar do programador que fez aquele código, que construiu aquela funcionalidade. Ou seja, hackear sistemas significa hackear pessoas, hackear formas de pensar.

Torna-se necessário um amadurecimento urgente na forma como o software é desenvolvido, bem como na qualidade do código produzido. É necessário que a segurança dos dados dos usuários e até mesmo a segurança deles mesmos seja a prioridade ao se construir um software. Isso é algo imprescindível para que se tenha sistemas verdadeiramente seguros, que possam ser cada vez mais úteis e trazer cada vez mais benefícios à sociedade.

5 Considerações Finais

Com este trabalho, torna-se evidente que a Nuvem, em certas circunstâncias, não oferece a segurança presumida por muitas empresas e usuários, revelando situações práticas que poderiam comprometer a segurança de APIs, usuários e sistemas. Essa análise concreta destaca lacunas potenciais e reforça a necessidade de considerar cuidadosamente a segurança nesse ambiente.

Portanto, compreender cada vez mais o funcionamento desse ecossistema é crucial, especialmente a interconexão dos diversos serviços disponíveis para formar essa estrutura complexa. É essencial identificar as falhas existentes atualmente e desenvolver métodos ágeis para detectar novas vulnerabilidades. Essas medidas não apenas fortalecerão a segurança para empresas e usuários, mas também mitigarão potenciais ataques que poderiam ter efeitos desastrosos no futuro, promovendo assim uma maior confiabilidade nesse cenário em constante evolução.

Há uma série de direções que poderiam expandir e complementar o escopo deste estudo. Uma área promissora seria a realização de uma análise comparativa de eficiência entre as ferramentas utilizadas, investigando profundamente suas capacidades de detecção de vulnerabilidades e sua adaptabilidade a diferentes ambientes. Essa análise detalhada poderia fornecer insights valiosos para orientar a escolha e o aprimoramento das ferramentas de segurança em ambientes de computação em nuvem.

Além disso, um campo de pesquisa adicional seria o estudo das técnicas de aplicação dessas ferramentas específicas. Compreender como extrair o máximo potencial de cada ferramenta, ajustando suas configurações e estratégias de utilização, poderia otimizar a eficácia dos testes de segurança, garantindo uma cobertura mais abrangente e precisa.

Outro caminho interessante seria realizar uma investigação para mensurar os possíveis impactos de cada falha identificada, caso fossem exploradas. Desenvolver métricas para avaliar o potencial de danos de cada vulnerabilidade poderia contribuir significativamente para a definição de um índice de segurança para as APIs. Essa abordagem

quantitativa poderia oferecer uma visão mais precisa do nível de risco associado a cada falha e direcionar os esforços de segurança de maneira mais eficiente.

Explorar esses caminhos de pesquisa adicionais não apenas ampliaria o conhecimento sobre segurança em ambientes de computação em nuvem, mas também poderia fornecer ferramentas e métricas práticas para melhorar a segurança das APIs, contribuindo assim para um ambiente digital mais resiliente e confiável.

Bibliografia

AHMET, E.; GÜNGÖR, M. O. The impact of meltdown and spectre attacks. *International Journal of Multidisciplinary Studies and Innovative Technologies*, v. 3, n. 1, p. 38–43, 2019.

ALMUTAIRY, N. M.; AL-SHQEERAT, K. H.; HAMAD, H. A. A. A taxonomy of virtualization security issues in cloud computing environments. *Indian Journal of Science and Technology*, MISC, v. 12, n. 3, p. 1–19, 2019.

ALZAIN, M. A.; PARDEDE, E.; SOH, B.; THOM, J. A. Cloud computing security: from single to multi-clouds. In: IEEE. *2012 45th Hawaii International Conference on System Sciences*. [S.l.], 2012. p. 5490–5499.

ARIFFIN, M. A. M.; IBRAHIM, M. F.; KASIRAN, Z. Api vulnerabilities in cloud computing platform: Attack and detection. *International Journal of Engineering Trends and Technology*, v. 1, p. 8–14, 2020.

BASU, S.; BARDHAN, A.; GUPTA, K.; SAHA, P.; PAL, M.; BOSE, M.; BASU, K.; CHAUDHURY, S.; SARKAR, P. Cloud computing security challenges & solutions-a survey. In: IEEE. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.], 2018. p. 347–356.

BISONG, E. *Building machine learning and deep learning models on Google cloud platform*. [S.l.]: Springer, 2019.

BOSS, G.; MALLADI, P.; QUAN, D.; LEGREGNI, L.; HALL, H. Cloud computing. *IBM white paper*, http:,/downl0ad. boulder. ibm. com/ibmdl/pub/software/dw/wes/hipods/-Cloud . . . , v. 321, p. 224–231, 2007.

BOX, D.; EHNEBUSKE, D.; KAKIVAYA, G.; LAYMAN, A.; MENDELSOHN, N.; NIELSEN, H. F.; THATTE, S.; WINER, D. *Simple object access protocol (SOAP) 1.1*. 2000.

CARGA útil (computação). Wikimedia Foundation, 2022. Disponível em: ⟨https://pt.wikipedia.org/wiki/Carga_%C3%BAtil_(computa%C3%A7%C3%A3o)⟩.

CERULLO, F. E. Owasp top 10 2009. In: SPRINGER. *Iberic Web Application Security Conference*. [S.l.], 2009. p. 19–19.

CIOTTI, M.; CICCOZZI, M.; TERRINONI, A.; JIANG, W.-C.; WANG, C.-B.; BERNARDINI, S. The covid-19 pandemic. *Critical reviews in clinical laboratory sciences*, Taylor & Francis, v. 57, n. 6, p. 365–388, 2020.

CONTI, M.; DRAGONI, N.; LESYK, V. A survey of man in the middle attacks. *IEEE communications surveys & tutorials*, IEEE, v. 18, n. 3, p. 2027–2051, 2016.

CRAIGEN, D.; DIAKUN-THIBAULT, N.; PURSE, R. Defining cybersecurity. *Technology Innovation Management Review*, v. 4, n. 10, 2014.

EBERT, C.; GALLARDO, G.; HERNANTES, J.; SERRANO, N. Devops. *Ieee Software*, IEEE, v. 33, n. 3, p. 94–100, 2016.

GOYAL, S. Public vs private vs hybrid vs community-cloud computing: a critical review. *International Journal of Computer Network and Information Security*, Modern Education and Computer Science Press, v. 6, n. 3, p. 20–29, 2014.

JACOBSON, D.; BRAIL, G.; WOODS, D. *APIs: A strategy guide*. [S.l.]: "O'Reilly Media, Inc.", 2012.

JANSEN, W.; GRANCE, T. Guidelines for security and privacy in pubic cloud. *Draft Special Publication*, p. 800–144.

JING, X.; JIAN-JUN, Z. A brief survey on the security model of cloud computing. In: IEEE. *2010 ninth international symposium on distributed computing and applications to business, engineering and science*. [S.l.], 2010. p. 475–478.

JOHNSTON, C. In-platform ci/cd. In: *Advanced Platform Development with Kubernetes*. [S.l.]: Springer, 2020. p. 117–152.

KADAM, Y. Security issues in cloud computing a transparent view. *International Journal of Computer Science Emerging Technology*, v. 2, n. 5, p. 316–322, 2011.

KHARRAZ, A.; ROBERTSON, W.; BALZAROTTI, D.; BILGE, L.; KIRDA, E. Cutting the gordian knot: A look under the hood of ransomware attacks. In: SPRINGER. *International conference on detection of intrusions and malware, and vulnerability assessment*. [S.l.], 2015. p. 3–24.

MARTINS, A. F.; FIGUEIREDO, M. A.; AGUIAR, P. M.; SMITH, N. A.; XING, E. P. Alternating directions dual decomposition. *arXiv preprint arXiv:1212.6550*, 2012.

MASSE, M. *REST API design rulebook: designing consistent RESTful web service interfaces*. [S.l.]: "O'Reilly Media, Inc.", 2011.

MELL, P.; GRANCE, T. et al. The nist definition of cloud computing. Computer Security Division, Information Technology Laboratory, National . . . , 2011.

METSCH, T.; EDMONDS, A.; PARÁK, B. Open cloud computing interface-infrastructure. In: *Standards Track, no. GFD-R in The Open Grid Forum Document Series, Open Cloud Computing Interface (OCCI) Working Group, Muncie (IN)*. [S.l.: s.n.], 2010.

MIRSKY, Y.; MAHLER, T.; SHELEF, I.; ELOVICI, Y. {CT-GAN}: Malicious tampering of 3d medical imagery using deep learning. In: *28th USENIX Security Symposium (USENIX Security 19)*. [S.l.: s.n.], 2019. p. 461–478.

MOHAMMED, C. M.; ZEEBAREE, S. R. et al. Sufficient comparison among cloud computing services: Iaas, paas, and saas: A review. *International Journal of Science and Business*, IJSAB International, v. 5, n. 2, p. 17–30, 2021.

OWASP API Security Top 10 — apisecurity.io. ⟨https://apisecurity.io/encyclopedia/content/owasp/owasp-api-security-top-10.htm⟩. [Accessed 28-Dec-2022].

PALMA, F.; GONZALEZ-HUERTA, J.; MOHA, N.; GUÉHÉNEUC, Y.-G.; TREMBLAY, G. Are restful apis well-designed? detection of their linguistic (anti) patterns. In: SPRINGER. *International Conference on Service-Oriented Computing*. [S.l.], 2015. p. 171–187.

PALMA, F.; GONZALEZ-HUERTA, J.; MOHA, N.; GUÉHÉNEUC, Y.-G.; TREMBLAY, G. Service-oriented computing: 13th. In: *International Conference, ICSOC*. [S.l.: s.n.], 2015. p. 16–19.

PETRILLO, F.; MERLE, P.; MOHA, N.; GUÉHÉNEUC, Y.-G. Are rest apis for cloud computing well-designed? an exploratory study. In: SPRINGER. *International Conference on Service-Oriented Computing*. [S.l.], 2016. p. 157–170.

QIAN, L.; LUO, Z.; DU, Y.; GUO, L. Cloud computing: An overview. In: SPRINGER. *IEEE international conference on cloud computing*. [S.l.], 2009. p. 626–631.

RANI, D.; RANJAN, R. K. A comparative study of saas, paas and iaas in cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, v. 4, n. 6, 2014.

REDHATC. *An architect's guide to APIs: SOAP, REST, GraphQL, and gRPC*. 2020. Disponível em: ⟨https://www.redhat.com/architect/apis-soap-rest-graphql-grpc⟩.

RODRÍGUEZ, C.; BAEZ, M.; DANIEL, F.; CASATI, F.; TRABUCCO, J. C.; CANALI, L.; PERCANNELLA, G. Rest apis: a large-scale analysis of compliance with principles and best practices. In: SPRINGER. *International conference on web engineering*. [S.l.], 2016. p. 21–39.

SCHWABER, K. Scrum development process. In: *Business object design and implementation*. [S.l.]: Springer, 1997. p. 117–134.

SEFRAOUI, O.; AISSAOUI, M.; ELEULDJ, M. et al. Openstack: toward an open-source solution for cloud computing. *International Journal of Computer Applications*, v. 55, n. 3, p. 38–42, 2012.

SMITH, E. L. J. A. L. Z. M. The hidden costs of cybercrime. In: *The Hidden Costs of Cybercrime*. [S.l.]: McAfee, 2020.

SUBRAMANIAN, N.; JEYARAJ, A. Recent security challenges in cloud computing. *Computers & Electrical Engineering*, Elsevier, v. 71, p. 28–42, 2018.

SURYATEJA, P. Threats and vulnerabilities of cloud computing: a review. *International Journal of Computer Sciences and Engineering*, v. 6, n. 3, p. 297–302, 2018.

THÖNES, J. Microservices. *IEEE software*, IEEE, v. 32, n. 1, p. 116–116, 2015.

WAMBA, S. F.; KAMDJOUG, J. R. K.; BAWACK, R. E.; KEOGH, J. G. Bitcoin, blockchain and fintech: a systematic review and case studies in the supply chain. *Production Planning & Control*, Taylor & Francis, v. 31, n. 2-3, p. 115–142, 2020.

YERGEAU, F.; BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M.; MALER, E. Extensible markup language (xml) 1.0. *W3C Recommendation*, v. 4, p. 220, 2004.

6 Apêndices

Apêndice 1 - Vulnerabilidades detectadas durante o estudo juntamente um identificador de qual foi a metodolgia aplicada que identificou a vulnerabilidade, qual é o CWE associado e quais ferramentas foram utilizadas no processo.

Host	Endpoint	Metodologia	CWEs Associados	Ferramentas Utilizadas
api1	/documentos/..../index.php?directory=../../	WSTG-ATHZ-01	CWE-22	Navegador + Google Dork + Postman
api2	/arquivos/.../php.ini	WSTG-ATHZ-01	CWE-22	Navegador + Google Dork + Postman
api2	/.../php.ini	WSTG-ATHZ-01	CWE-22	Navegador + Google Dork + Postman
api3	/.../database/mysql/	WSTG-CONF-02	CWE-16/CWE-703 / CWE-254	Navegador + Google Dork + Postman
api4	/app_r...$_p swd$/	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork + Postman
api5	/.../reminder/exec/	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork + Postman
api6	/c.../app$_L ogin$/	WSTG-CRYP-01	CWE-311	Navegador + Google Dork + Postman
api7	/form/admin.php?mod=...&func=...	WSTG-IDNT-04	CWE-200/CWE-201	Google Dork + Postman
api8	/	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Navegador + Google Dork + Postman
api9	/	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Navegador + Google Dork + Postman
api10	/	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Navegador + Google Dork + Postman
api11	/login	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork + Postman
api12	/wp-json/	WSTG-CONF-02	CWE-16/CWE-703/CWE-254	Navegador + Google Dork + Postman
api13	/ticket$_a nexo$/.../../...pdf	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api14	/ticket$_a nexo$/.../.../...pdf	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api15	/.../.../compra...pdf	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman

api16	/wp-content/uploads/.../01/...2020-LOTES-URBANIZADO.pdf	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api17	/public/.../doc/20190722....pdf	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api18	/?urlgo=https://.../compartilhe/7	WSTG-IDNT-047	CWE-200/CWE-2017	Navegador + Google Dork + Postman
api19	/pms/.../users.txt	WSTG-INFO-03	CWE-522/CWE-256/CWE-522/CWE-330	Navegador + Google Dork + Postman
api20	/media/.../attachments/.../files/ta...es.txt	WSTG-INFO-03	CWE-522/CWE-256/CWE-522/CWE-330	Navegador + Google Dork + Postman
api21	/.../page-detalhes.php?cod=...	WSTG-INPV-05	CWE-89	Navegador + Google Dork + Postman
api22	/awstats/.../aws...19.txp.pt.txt	WSTG-INFO-03	CWE-522/CWE-256/CWE-522/CWE-330	Navegador + Google Dork + Postman
api23	/.../data/aws..015.gm..ia.pt.txt	WSTG-INFO-03	CWE-522/CWE-256/CWE-522/CWE-330	Navegador + Google Dork + Postman
api24	/nvidia/.../3X/../f/7fb...2d.log	WSTG-INFO-03	CWE-522/CWE-256/CWE-522/CWE-330	Navegador + Google Dork + Postman
api25	/.../uploads/.../original/...bc09497c.log	WSTG-INFO-03	CWE-522/CWE-256/CWE-522/CWE-330	Navegador + Google Dork + Postman
api26	/uploads/.../vQ39...CV3bXm.log	WSTG-INFO-03	CWE-522/CWE-256/CWE-522/CWE-330	Navegador + Google Dork + Postman
api27	/u/...?username=test2...email=t...fum.com	WSTG-INFO-01	CWE-200/CWE-2000	Navegador + Burp suite
api28	/../?email=timl...	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Burp suite
api29	/.../forgot$_{p}assword$	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Burp suite
api29	/session/...	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Burp suite

api30	/.../v1/accounts/...	WSTG-IDNT-04	CWE-601	Android device + Burp Suite + Android ADB
api31	/.../v1/orders/...?status=ALL&page=0&...	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Android device + Burp Suite + Android ADB
api32	/recommende...	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Android device + Burp Suite + Android ADB
api33	/.../echo.php?ip=xxx.xxx.xx.xxx	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Android device + Burp Suite + Android ADB
api34	/	WSTG-ATHZ-02	CWE-287/CWE-285/CWE-200	Android device + Burp Suite + Android ADB
api35	/forgot-password	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork + Postman
api36	/.../boot../.../1b1f...1b4e99	WSTG-INFO-01	CWE-522/CWE-256/CWE-542/CWE-320	Navegador + Google Dork + Postman
api37	/.../Projet../../blob/1d0.../.env	WSTG-INFO-03	CWE-522/CWE-256/CWE-522/CWE-330	Navegador + Google Dork + Postman
api38	/.env	WSTG-INFO-01	CWE-522/CWE-256/CWE-320	Navegador + Google Dork + Postman
api39	/uploads/...	WSTG-INFO-01	CWE-522/CWE-256/CWE-542/CWE-320	Navegador + Google Dork + Postman
api40	/wp-conf....txt	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api41	/css/.../jumping/vben....php.txt	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api42	/file..513/pthon-...3.5.4.txt	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api43	/?id=C03F....D96%216455&cid=C03..6	WSTG-INFO-01	CWE-522/CWE-256/CWE-542/CWE-320	Navegador + Google Dork + Postman

api44	/?id=455C5F...B8	WSTG-INFO-01	CWE-522/CWE-256/CWE-542/CWE-320	Navegador + Google Dork + Postman
api45	/history/94665...6d/SQL/	WSTG-ATHN-02	CWE-287/CWE-532	Navegador + Google Dork + Postman
api46	/jetst.../..logs/pull/...ert-manager/.../pull-c..-16/1.../...log.txt	WSTG-CONF-11	CWE-16	Navegador + Google Dork + Postman
api47	/...logs/	WSTG-CONF-11	CWE-16	Navegador + Google Dork + Postman
api48	/	WSTG-IDNT-04	CWE-200/CWE-201	Android device + Burp Suite + Android ADB
api49	/arquivos/...Acessos_...$xlsx$	WSTG-CONF-11	CWE-16	Navegador + Google Dork + Postman
api50	/.../...$_{C}orporativo.xlsx$	WSTG-CONF-11	CWE-16	Navegador + Google Dork + Postman
api51	/legacy/../..-files/a...$_{u}nify$/...$update_{2}.sql$	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api52	/fi/com.../2../07/18..-01-25$_{t}ransferencia - pix - ...comercio - eletronico..{}_{6}9.PDF$	WSTG-CONF-04	CWE-200/CWE-201/CWE-256	Navegador + Google Dork + Postman
api53	/	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Android device + Burp Suite + Android ADB
api54	/wp-login.php?action=lostpassword	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork + Postman
api55	/forum/sign-up/?redirect$_{t}o$ = $http\%3A\%2F\%2Fwww.s...z...gov.br\%2Fforum\%2F$	WSTG-CRYP-01	CWE-311	Navegador + Google Dork + Postman
api56	/wp-admin/	WSTG-CLNT-04	CWE-601	Navegador + Google Dork + Postman
api57	/redirect.php?url=http://www...com/main/....php?lang=en	WSTG-CLNT-04	CWE-601	Navegador + Google Dork + Postman
api58	/en/page/redirect?url=https://www....org/en/.../mission...	WSTG-CLNT-04	CWE-601	Navegador + Google Dork + Postman

api59	/v1/redirect?url=https...api$_key$ = $92cc8c960...82b3\&site_id = 52a...8a89e\&source = https...1698956753 - 0 - gaNycGzNDxA$	WSTG-CLNT-04	CWE-601	Navegador + Google Dork + Postman
api60	/	WSTG-CRYP-01	CWE-311	Navegador + Google Dork + Postman
api61	/wp-login.php	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork + Postman
api62	/	WSTG-CRYP-01	CWE-311	Navegador + Google Dork + Postman
api63	/	WSTG-CONF-10	CWE-16	Subjack + Subfinder + Scriptpython
api64	/	WSTG-CONF-10	CWE-16	Subjack + Subfinder + Scriptpython
api65	/	WSTG-CONF-10	CWE-16	Subjack + Subfinder + Scriptpython
api66	/	WSTG-CONF-10	CWE-16	Subjack + Subfinder + Scriptpython
api67	/	WSTG-CONF-10	CWE-16	Subjack + Subfinder + Scriptpython
api68	/	WSTG-CONF-10	CWE-16	Subjack + Subfinder + Scriptpython
api69	/graphql	WSTG-BUSL-01	CWE-20	Android device + Burp Suite + Android ADB
api70	/	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Android device + Burp Suite + Android ADB
api71	/	WSTG-INPV-05	CWE-89	Android device + Burp Suite + Android ADB
api72	/	WSTG-BUSL-01	CWE-287	Android device + Burp Suite + Android ADB
api73	/users/4062.../	WSTG-ATHZ-02	CWE-287/CWE-285/CWE-200	Android device + Burp Suite + Android ADB
api74	/idm/.../generateCode	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork + Postman

api75	/Blac...HostServi...Service.1...1.1.zip	WSTG-INFO-01	CWE-522/CWE-256/CWE-542/CWE-320	Navegador + Google Dork+Burp Suite
api76	/cat.php?id=%27	WSTG-INPV-05	CWE-89	Navegador + Google Dork+Burp Suite
api77	/api/.../public/.../authentication/acesskey/	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork+Burp Suite
api78	/jour.../.../admin	WSTG-ATHN-07	CWE-521	Navegador + Google Dork+Burp Suite
api79	/journal/.../user/.../6551...6c93	WSTG-ATHN-04	CWE-285	Navegador + Google Dork+Burp Suite
api80	/.../perii/.../payment$_{i}nfo$	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Navegador + Google Dork+Burp Suite
api81	/forgot-password	WSTG-IDNT-04	CWE-200/CWE-201	Navegador
api82	/info.php	WSTG-CONF-02	CWE-16/CWE-703/CWE-254	Navegador
api83	/.../editContrato/3...	WSTG-CONF-02	CWE-16/CWE-703/CWE-254	Navegador + Google Dork+Burp Suite
api84	/u.../concrete.../conc../src/File/StorageLocation/.../	WSTG-CONF-02	CWE-16/CWE-703/CWE-254	Navegador + Google Dork+Burp Suite
api85	/external?href=https://www.../bi.../redirect.php?goto=http://...	WSTG-CLNT-04	CWE-601	Navegador + Google Dork+Burp Suite
api86	/wp-content/uploads/mc....log	WSTG-CONF-11	CWE-16	Navegador + Google Dork+Burp Suite
api87	/forgot-password.php	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork+Burp Suite
api88	/Etudes/$_{d}$.../$WS...LOG$	WSTG-CONF-11	CWE-16	Navegador + Google Dork+Burp Suite
api89	/sho....php?id=	WSTG-INPV-05	CWE-89	Navegador + Google Dork+Burp Suite
api90	/var/www/html/	WSTG-ATHZ-01	CWE-22	Navegador + Google Dork+Burp Suite

api91	/ s.../d...html	WSTG-IDNT-04	CWE-200/CWE-201	Navegador + Google Dork+Burp Suite
api92	/	WSTG-CONF-02	CWE-16/CWE-703/CWE-254	Navegador + Google Dork+Burp Suite
api93	/human-.../pa.../sc.../202...01$_m$*i...aph...*/*...hal.log*	WSTG-CONF-11	CWE-16	Navegador + Google Dork+Burp Suite
api94	/	WSTG-INFO-02	CWE-2000/CWE-215/CWE-200	Navegador + Google Dork+Burp Suite
api95	/.../d/e/1FA.../viewan...	WSTG-CONF-11	CWE-16	Navegador + Google Dork+Burp Suite

Tabela 6.1: Resultados do Estudo Exploratório