

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Tarcísio Júnio de Araujo Resende

**Segurança em nuvem - Um estudo de caso em
AWS para proteção de uma aplicação Web**

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Tarcísio Júnio de Araujo Resende

**Segurança em nuvem - Um estudo de caso em AWS para
proteção de uma aplicação Web**

Trabalho de conclusão de curso apresentado
à Faculdade de Computação da Universidade
Federal de Uberlândia, como parte dos requi-
sitos exigidos para a obtenção título de Ba-
charel em Ciência da Computação.

Orientador: Prof. Dr. Rodrigo Sanches Miani

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

Tarcísio Júnio de Araujo Resende

Segurança em nuvem - Um estudo de caso em AWS para proteção de uma aplicação Web

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 15 de Maio de 2025:

Prof. Dr. Rodrigo Sanches Miani
Orientador

Prof. Dr. Luís Fernando Faina
Professor

Prof. Dr. Maria Adriana Vidigal de Lima
Professor

Uberlândia, Brasil
2025

Resumo

Este trabalho teve como objetivo analisar de forma prática três classes de ataques cibernéticos que afetam aplicações web em ambientes de computação em nuvem. Para tanto, foram selecionados cenários de ataques de negação de serviço (DoS), injeção de SQL (SQLi) e *Server-Side Request Forgery* (SSRF) em instâncias AWS EC2 configuradas sem proteções específicas. A metodologia consistiu em simular cada tipo de ataque em um ambiente controlado, coletando métricas de disponibilidade, uso de recursos e registros de *logs* do sistema. Em seguida, foram implementadas contramedidas nativas da AWS, incluindo *Application Load Balancer* (ALB), *Web Application Firewall* (WAF) e configuração de *Instance Metadata Service Version 2* (IMDSv2). Nos testes de DoS, aplicou-se regras de *rate limiting* e desafios *CAPTCHA* no WAF para limitar requisições por origem IP. No cenário de SQLi, utilizou-se o conjunto de regras gerenciadas *AWSManagedRulesSQLiRuleSet* para bloquear tentativas de injeção sem alterações no código-fonte. Para SSRF, combinou-se IMDSv2 com regras específicas do WAF, impedindo acesso ao serviço de metadados e protegendo credenciais temporárias. Os resultados mostraram que o AWS WAF bloqueou eficientemente tanto o tráfego excessivo de DoS quanto as tentativas de injeção de código e SSRF. A análise de *logs* confirmou a detecção e o bloqueio das requisições maliciosas em todos os cenários testados. Apesar da eficácia das soluções AWS, constatou-se que boas práticas de desenvolvimento, como validação de entradas e uso de consultas parametrizadas, permanecem essenciais. A configuração de privilégios mínimos nas *roles* também é fundamental para limitar o impacto em caso de comprometimento. Por fim, este estudo reforça que a nuvem oferece poderosas ferramentas de segurança, mas é preciso saber configurá-las de forma correta, já que as proteções não são automaticamente habilitadas.

Palavras-chave: Computação em nuvem, Segurança da informação, AWS, *Cloud*, WAF, DoS, *SQL Injection*, SSRF.

Lista de ilustrações

Figura 1 – Arquitetura inicial	21
Figura 2 – Site Web	22
Figura 3 – Gráfico de Utilização da instância	23
Figura 4 – Uso de recursos da máquina	23
Figura 5 – Site fora do ar	23
Figura 6 – Curl para o site	24
Figura 7 – Arquitetura da solução proposta	24
Figura 8 – Regra de <i>Rate Limit</i>	25
Figura 9 – Requisição bloqueada	25
Figura 10 – <i>Captcha</i>	26
Figura 11 – Arquitetura AWS	27
Figura 12 – Página de login	28
Figura 13 – Login realizado	29
Figura 14 – Regra <i>SQL Injection</i>	29
Figura 15 – Requisição bloqueada	30
Figura 16 – Log WAF	30
Figura 17 – Site resumidor de páginas	32
Figura 18 – Exemplo do funcionamento do site	33
Figura 19 – Testando IP Metadados	33
Figura 20 – Consultando nome da <i>Role</i>	34
Figura 21 – Obtendo credenciais	34
Figura 22 – Configurando IMDSv2	35
Figura 23 – Falha na obtenção de credenciais	36
Figura 24 – Arquitetura com WAF	36
Figura 25 – Log WAF	37

Lista de abreviaturas e siglas

ALB	<i>Application Load Balancer</i>
API	<i>Application Programming Interface</i>
AWS	<i>Amazon Web Services</i>
CAPTCHA	<i>Completely Automated Public Turing test to tell Computers and Humans Apart</i>
CIA	<i>Confidentiality, Integrity, Availability</i>
CSP	<i>Cloud Service Provider</i>
DDoS	<i>Distributed Denial of Service</i>
DoS	<i>Denial of Service</i>
EC2	<i>Elastic Compute Cloud</i>
HTTP	<i>HyperText Transfer Protocol</i>
HTTPS	<i>HyperText Transfer Protocol Secure</i>
IaaS	<i>Infrastructure as a Service</i>
IMDS	<i>Instance Metadata Service</i>
IMDSv1	<i>Instance Metadata Service version 1</i>
IMDSv2	<i>Instance Metadata Service version 2</i>
MySQL	<i>My Structured Query Language</i>
NIST	<i>National Institute of Standards and Technology</i>
OWASP	<i>Open Web Application Security Project</i>
PaaS	<i>Platform as a Service</i>
PHP	<i>PHP: Hypertext Preprocessor</i>
SaaS	<i>Software as a Service</i>
SGBD	Sistema Gerenciador de Banco de Dados
SLA	<i>Service Level Agreement (Acordo de Nível de Serviço)</i>

SQL	<i>Structured Query Language</i>
SQLi	<i>SQL Injection</i>
SSRF	<i>Server-Side Request Forgery</i>
WAF	<i>Web Application Firewall</i>

Sumário

1	INTRODUÇÃO	9
1.1	Objetivos	10
1.1.1	Objetivo geral	10
1.1.2	Objetivos específicos	10
1.2	Justificativa	10
1.3	Organização da monografia	11
2	REFERENCIAL TEÓRICO	12
2.1	Segurança da informação	12
2.2	Computação em nuvem	13
2.3	Amazon Web Services (AWS)	14
2.4	Ataques cibernéticos	16
2.4.1	Negação de serviço	16
2.4.2	SQL injection	16
2.4.3	<i>Server-side request forgery</i>	17
2.5	Trabalho correlatos	18
3	DESENVOLVIMENTO	20
3.1	Metodologia	20
3.1.1	Descrição do cenário de testes	20
3.1.2	Cenário de ataque	20
3.1.3	Implementação das medidas de segurança	21
3.1.4	Avaliação e análise dos resultados	21
3.2	Ataque de negação de serviço - DoS	21
3.2.1	Descrição do cenário de testes	21
3.2.2	Cenário de ataque	22
3.2.3	Implementação das medidas de segurança	24
3.2.4	Avaliação e análise dos resultados	26
3.3	<i>SQL injection</i> - <i>SQLi</i>	27
3.3.1	Descrição do cenário de testes	27
3.3.2	Cenário de ataque	28
3.3.3	Implementação das medidas de segurança	29
3.3.4	Avaliação e análise dos resultados	31
3.4	<i>Server-side request forgery</i> - <i>SSRF</i>	31
3.4.1	Descrição do cenário de testes	31
3.4.2	Cenário de ataque	32

3.4.3	Implementação das medidas de segurança	35
3.4.4	Avaliação e análise dos resultados	37
4	CONCLUSÃO	39
	REFERÊNCIAS	41

1 Introdução

Desde os primórdios da computação, a busca por soluções eficientes para armazenamento, processamento e distribuição de dados tem sido constante. A evolução tecnológica trouxe a promessa de flexibilidade, escalabilidade e acessibilidade por meio da computação em nuvem.

De acordo com (BUYAYA et al., 2009), o sistema em nuvem é um modelo de computação paralela e distribuída, composto por uma coleção de computadores interconectados e virtualizados, provisionados dinamicamente e apresentados como um ou mais recursos computacionais unificados. Esse design permite escalabilidade sob demanda e padronização de recursos, atendendo a diferentes requisitos de desempenho e custo.

Já o *The National Institute of Standards and Technology* (NIST) (MELL; GRANCE, 2011) define computação em nuvem como "um modelo que permite acesso conveniente e sob demanda à rede, possibilitando o uso compartilhado de uma variedade de recursos computacionais configuráveis (como redes, servidores, aplicativos e serviços de armazenamento), que podem ser provisionados e liberados rapidamente, com mínimo esforço de gestão ou interação com o provedor de serviços.". Essa definição ressalta a agilidade e a flexibilidade inerentes à computação em nuvem, fundamentais para suprir demandas variáveis dos usuários.

A computação em nuvem evoluiu de um conceito de negócio promissor para se tornar um dos segmentos de crescimento mais rápido da indústria de TI (POPOVIĆ; HOCENSKI, 2010). Com o crescimento, surgiram uma grande variedade de provedores de nuvem (*Cloud Providers*) (TASNIM et al., 2022).

Os Provedores de Serviços em Nuvem (*Cloud Service provider* CSP) são fornecedores que alugam aos seus clientes serviços em nuvem que são dinamicamente utilizados com base na demanda dos clientes (ALMISHAL; YOUSSEF, 2014). Esses provedores disponibilizam recursos como máquinas virtuais, armazenamento e redes sob demanda, permitindo que organizações ajustem sua infraestrutura sem grandes investimentos iniciais.

Dentre os provedores de serviços em nuvem, destaca-se a Amazon Web Services (AWS). Fundada em 2006, a AWS rapidamente se tornou a líder indiscutível no mercado de computação em nuvem, oferecendo uma ampla gama de serviços, desde infraestrutura básica até ferramentas de inteligência artificial e aprendizado de máquina. Uma pesquisa de 2022 (TASNIM et al., 2022) aponta que a AWS possuía 32% de participação do mercado global de nuvem, superando seus principais concorrentes, como Microsoft Azure e Google Cloud Platform (GCP).

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo deste trabalho é analisar e testar cenários de ataque em aplicações web hospedadas na nuvem da AWS. Para isso, o estudo analisará as possíveis soluções de mitigação, visando aprimorar a segurança e a confiabilidade desses sistemas por meio do uso das tecnologias da AWS disponíveis para cada caso.

1.1.2 Objetivos específicos

- Analisar o cenário atual de segurança em aplicativos web e os desafios específicos enfrentados por organizações que utilizam infraestrutura de nuvem, com ênfase na AWS.
- Identificar e testar serviços de segurança disponíveis na AWS para mitigação de ameaças, destacando suas funcionalidades, limitações e melhores práticas de configuração.
- Realizar um estudo de caso prático para aplicar os conceitos e as soluções de segurança da AWS em um ambiente de desenvolvimento de aplicativos web, avaliando sua eficácia na detecção e prevenção de ameaças.

1.2 Justificativa

A rápida adoção de serviços em nuvem tem sido acompanhada por um aumento alarmante na quantidade de incidentes de segurança. Segundo a pesquisa *Cloud Security Report* ([CHECKPOINT, 2024a](#)) de 2024, 61% das empresas relataram ter experimentado algum tipo de incidente de segurança em seus ambientes de nuvem nos últimos 12 meses, um aumento significativo em comparação aos 24% registrados em 2023. Este crescimento substancial destaca a escalada das ameaças cibernéticas e reforça a necessidade de estratégias de segurança eficientes e robustas para proteger dados e aplicações na nuvem.

Os desafios de segurança em um ambiente de nuvem são complexos e diversos, e sua natureza remota traz desafios únicos em segurança ([CHECKPOINT, 2024b](#)). A infraestrutura compartilhada, a necessidade de gerenciar identidades e acessos de maneira eficaz, e a proteção contra ameaças internas e externas representam apenas algumas das dificuldades que as organizações enfrentam. A gestão adequada dessas áreas é crucial para garantir a integridade e a confidencialidade dos dados armazenados na nuvem.

A segurança deve ser abordada de forma integrada, combinando políticas, processos e tecnologias para mitigar riscos. No entanto, segundo a pesquisa ([CHECKPOINT,](#)

2024a), apenas 21% das empresas priorizam estratégias de prevenção para impedir ataques antes que eles ocorram. Esse dado ressalta a necessidade urgente de um maior investimento em medidas preventivas e em uma cultura de segurança mais proativa nas organizações.

Os prejuízos financeiros ocasionados pelo *cybercrime* continuam aumentando. Em 2015, foram estimados em 3 trilhões de dólares, com previsões de atingir 9,5 trilhões de dólares em 2024 e 10,5 trilhões de dólares em 2025 (VENTURES, 2023). Esses números refletem a crescente dependência de tecnologias digitais, incluindo serviços de nuvem, na economia global. Diante dessa escalada de prejuízos associados a incidentes de segurança, torna-se necessário implementar medidas de segurança em ambientes de nuvem para mitigar esses riscos financeiros e proteger os ativos digitais das organizações.

1.3 Organização da monografia

Esta monografia está organizada em quatro capítulos principais. O Capítulo 1, "Introdução", contextualiza o tema da segurança em nuvem, apresenta definições fundamentais, os objetivos geral e específicos e a justificativa do estudo. O Capítulo 2, "Referencial Teórico", revisa os conceitos de segurança da informação, computação em nuvem e os serviços da AWS, detalha os ataques de DoS, *SQL Injection* e SSRF, e analisa os trabalhos correlatos. No Capítulo 3, "Desenvolvimento", descreve-se a metodologia adotada, o ambiente e os cenários de testes, a simulação dos ataques e a implementação das contramedidas na AWS, além da avaliação e discussão dos resultados obtidos. Finalmente, o Capítulo 4, "Conclusão", sintetiza as principais contribuições do trabalho, aponta suas limitações e sugere direções para trabalhos futuros.

2 Referencial teórico

Neste capítulo, apresentam-se os fundamentos teóricos que embasam este trabalho, iniciando pelos pilares da segurança da informação e avançando pelos modelos de computação em nuvem, com ênfase nos serviços da AWS que compõem o ambiente de estudo. Em seguida, são examinadas três classes de ataques cibernéticos — negação de serviço, injeção de SQL e SSRF — e, por fim, discutem-se trabalhos correlatos a esta monografia.

2.1 Segurança da informação

A segurança da informação é um campo essencial da ciência da computação, focado na proteção de ativos informacionais contra ameaças que possam comprometer sua confidencialidade, integridade e disponibilidade. Segundo (STALLINGS; BROWN, 2014), esses três pilares — frequentemente referidos pela sigla CIA (*Confidentiality, Integrity, Availability*) — constituem os objetivos fundamentais da segurança da informação.

- Confidencialidade diz respeito à proteção do sigilo da informação, assegurando que somente pessoas, entidades ou processos devidamente autorizados tenham acesso aos dados sensíveis (International Organization for Standardization; International Electrotechnical Commission, 2018). Em outras palavras, a confidencialidade visa prevenir a divulgação ou exposição de informações a quem não possui permissão, salvaguardando segredos comerciais, dados pessoais e outras informações críticas contra espionagem ou vazamentos (International Organization for Standardization; International Electrotechnical Commission, 2018).

- Integridade refere-se à exatidão e completude das informações, assegurando que os dados não sejam alterados de forma indevida ou destruídos acidentalmente ou intencionalmente (International Organization for Standardization; International Electrotechnical Commission, 2018). Ou seja, uma perda de integridade consiste na modificação ou destruição não autorizada de informações (STALLINGS; BROWN, 2014). Garantir a integridade envolve controles como mecanismos de *hash* e assinatura digital, trilhas de auditoria, controles de versão e políticas de autorização rigorosas para edição ou exclusão de dados (STALLINGS; BROWN, 2014).

- Disponibilidade é a propriedade que assegura que as informações e os sistemas estejam acessíveis e utilizáveis sempre que necessários pelos usuários autorizados (International Organization for Standardization; International Electrotechnical Commission, 2018). Esse princípio garante que, no momento em que um dado ou serviço é requisitado

para uma finalidade legítima, ele possa ser obtido sem atrasos indevidos. A disponibilidade está intimamente ligada à continuidade de negócios, pois indisponibilidades podem interromper operações e causar prejuízos significativos ([Cybersecurity and Infrastructure Security Agency, 2024](#)).

Além desses princípios, [Stallings e Brown \(2014\)](#) destaca a importância de outros aspectos, como a autenticidade, que verifica a identidade dos usuários e sistemas, e a responsabilidade, que assegura que as ações dos usuários possam ser atribuídas de forma inequívoca. Esses conceitos são fundamentais para a implementação de políticas de segurança eficazes e para o desenvolvimento de sistemas resilientes a ameaças internas e externas.

2.2 Computação em nuvem

A computação em nuvem é um modelo que permite acesso amplo, conveniente e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços). Esses recursos são disponibilizados como serviços através da Internet e podem ser rapidamente provisionados e liberados com esforço mínimo de gerenciamento ou interação com o provedor de serviços ([MELL; GRANCE, 2011](#)).

O modelo de computação em nuvem apoia-se em cinco características essenciais, sendo elas autosserviço sob demanda, amplo acesso à rede, *pooling* de recursos, elasticidade rápida e serviço mensurado ([MELL; GRANCE, 2011](#)). Essas características permitem que organizações ajustem dinamicamente suas infraestruturas conforme a necessidade, obtendo maior eficiência de custos em várias situações ([ARMBRUST et al., 2010](#)).

Segundo ([MELL; GRANCE, 2011](#)), além das cinco características essenciais mencionadas, o modelo de computação em nuvem também é composto por três modelos de serviço:

- Infraestrutura como Serviço (IaaS), em que o provedor oferece recursos de infraestrutura como processamento, armazenamento e rede e outros recursos computacionais. Cabendo ao cliente gerenciar os sistemas operacionais, o armazenamento, as aplicações implantadas e, possivelmente, tendo controle limitado sobre determinados componentes de rede. Neste modelo o cliente não gerencia nem controla a infraestrutura de nuvem.
- Plataforma como Serviço (PaaS), onde é fornecida ao consumidor a capacidade de implantar na infraestrutura de nuvem aplicativos criados ou adquiridos pelo consumidor, criados usando linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor. Neste modelo, o cliente não gerencia ou controla a

infraestrutura de nuvem subjacente, incluindo rede, servidores, sistemas operacionais ou armazenamento.

- Software como Serviço (SaaS), em que aplicações completas são fornecidas pelo provedor, neste modelo o cliente não gerencia nem controla a infraestrutura em nuvem, somente possui controle sobre as aplicações implantadas e, possivelmente, sobre configurações do ambiente de hospedagem dessas aplicações.

Por último, ([MELL; GRANCE, 2011](#)) também divide a computação em nuvem em quatro modelos de implantação. Sendo eles:

- Nuvem Privada: Utilizada exclusivamente por uma organização. Pode ser gerenciada internamente ou por terceiros, e estar dentro ou fora das instalações da empresa.
- Nuvem Comunitária: Compartilhada por organizações com interesses em comum. A gestão pode ser feita pelas próprias organizações ou por terceiros.
- Nuvem Pública: Disponível para o público em geral, gerenciada por empresas, instituições acadêmicas ou órgãos governamentais, hospedada nas instalações do provedor de nuvem.
- Nuvem Híbrida: Combinação de dois ou mais modelos (privada, pública ou comunitária), permitindo integração e portabilidade de dados e aplicações entre eles.

2.3 Amazon Web Services (AWS)

Dentre os provedores de nuvem existentes, a Amazon Web Services (AWS) se destaca como uma das plataformas líderes mundiais em computação em nuvem, fornecendo um ecossistema amplo de serviços distribuídos em data centers globais. A AWS oferece infraestrutura e plataformas escaláveis sob demanda e seus serviços abrangem desde processamento, armazenamento e bancos de dados até redes e aprendizado de máquina, permitindo que organizações de diferentes portes hospedem, gerenciem e expandam suas operações online com segurança, eficiência e sem a necessidade de grandes investimentos iniciais ([PATIL, 2024](#)).

Entre os diversos serviços de infraestrutura oferecidos pela AWS, destacam-se mecanismos para distribuição de tráfego e proteção de aplicações web. O Amazon *Application Load Balancer* (ALB) é um componente de balanceamento de carga de camada 7 (camada de aplicação) que distribui automaticamente requisições HTTP/HTTPS entre múltiplas instâncias, contêineres ou IPs ([Amazon Web Services, 2024c](#)).

O ALB inspeciona o conteúdo de cada requisição (por exemplo, URL, cabeçalhos, *cookies*) e, com base em regras definidas, encaminha o tráfego para diferentes servidores

conforme necessário, suportando arquiteturas modernas como microsserviços e aplicações baseadas em contêineres. Isso possibilita uma melhor escalabilidade e tolerância a falhas, pois nenhuma instância individual fica sobrecarregada além de sua capacidade ([Amazon Web Services, 2024c](#)).

O balanceador de carga de aplicação contribui para a segurança ao centralizar pontos de entrada: ele pode assegurar que todas as conexões usem protocolos e cifragens TLS atualizados e pode integrar-se a outros serviços de segurança da AWS, como o AWS WAF, para inspecionar o tráfego malicioso antes mesmo de atingir as aplicações ([Amazon Web Services, 2024c](#)). Ou seja, o *Application Load Balancer* melhora tanto o desempenho quanto a proteção das aplicações em nuvem, ao distribuir adequadamente as cargas e permitir configurações de filtragem e roteamento avançadas no nível da aplicação.

Entre os vários serviços de segurança que a AWS oferece, está o AWS WAF (*Web Application Firewall*), um serviço gerenciado de *firewall* para aplicações web. O WAF permite monitorar e filtrar o tráfego HTTP/HTTPS direcionado às aplicações protegidas, com base em regras definidas que identificam padrões maliciosos ou indesejados ([Amazon Web Services, 2024b](#)). Com o WAF, é possível criar regras customizadas ou utilizar conjuntos de regras pré-configuradas para bloquear ataques comuns em aplicações web, como tentativas de *SQL Injection*, *Cross-Site Scripting* (XSS), *Server-Side Request Forgery*, entre outros ataques conhecidos ([Amazon Web Services, 2024b](#)). Por exemplo, pode-se configurar o WAF para negar requisições que contenham sequências suspeitas nos parâmetros (indicativas de código SQL ou *scripts*), ou bloquear endereços IP que excedam um limite de requisições por segundo (mitigando ataques de força bruta ou DoS na camada de aplicação) ([Amazon Web Services, 2024b](#)).

O serviço de *Web Application Firewall* da AWS integra-se facilmente a outros serviços da AWS, ele pode ser habilitado diretamente a um *Application Load Balancer*, distribuição CloudFront (rede de distribuição de conteúdo) ou API Gateway – de modo que todo o tráfego destinado a esses serviços passe pelos filtros do WAF antes de chegar à aplicação alvo. Dessa forma, o WAF atua como uma barreira de proteção adicional, aumentando a resiliência e segurança de aplicações em nuvem ao prevenir que requisições maliciosas explorem vulnerabilidades conhecidas ([PATIL, 2024](#)).

Além desses recursos, a AWS também utiliza o conceito de *Role* (função), que representa uma identidade com permissões definidas, usada para delegar acesso temporário a recursos da AWS sem a necessidade de credenciais permanentes. As *Roles* são amplamente utilizadas por serviços, usuários ou aplicações que precisam assumir permissões específicas para realizar ações dentro da nuvem de forma segura e controlada, seguindo os princípios de privilégio mínimo ([Amazon Web Services, 2024d](#)).

2.4 Ataques cibernéticos

Nesta seção, abordam-se três classes de ataques cibernéticos relevantes para este trabalho: negação de serviço (DoS), injeção de SQL (SQLi) e *Server-Side Request Forgery* (SSRF).

2.4.1 Negação de serviço

Negação de Serviço (DoS) caracteriza-se por ataques cujo objetivo é indisponibilizar um serviço, sistema ou rede para seus usuários legítimos, geralmente esgotando os recursos do alvo ou explorando vulnerabilidades que causem falhas ([Cybersecurity and Infrastructure Security Agency, 2024](#)). Numa situação de DoS típica, um atacante sobrecarrega o servidor ou a aplicação com um volume massivo de requisições ou tráfego inútil, acima da capacidade de processamento, até que o sistema fique lento ou totalmente fora do ar ([Cybersecurity and Infrastructure Security Agency, 2024](#)). Como consequência, usuários autorizados passam a não conseguir acessar o serviço (por exemplo, um site web ou API) durante o ataque, gerando interrupção de atividades e possivelmente prejuízos financeiros e reputacionais para a organização vítima ([Cybersecurity and Infrastructure Security Agency, 2024](#)).

Segundo ([Amazon Web Services, 2024a](#)), existem múltiplas técnicas de DoS: inundação de tráfego (*flooding*), exploração de protocolos (como ataques *SYN flood* no TCP), abusos de funcionalidades (por exemplo, envio de requisições complexas que consomem muito processamento) etc. Quando o ataque se origina de múltiplas máquinas distribuídas – frequentemente uma rede de *bots* previamente comprometidas – denomina-se DDoS (*Distributed DoS*), tornando ainda mais difícil de mitigar devido à quantidade de origens diferentes ([Amazon Web Services, 2024a](#)).

Para defender contra DoS/DDoS, empregam-se estratégias de filtragem de tráfego, limites de taxa (*rate limiting*), redes de distribuição de conteúdo e serviços de mitigação especializados, como o AWS Shield, que identificam e bloqueiam tráfego malicioso em escala antes que ele atinja os servidores principais ([PATIL, 2024](#)). Mesmo com contramedidas, ataques de negação de serviço distribuído continuam sendo uma ameaça persistente à disponibilidade, requerendo preparo e monitoramento constante por parte das equipes de segurança.

2.4.2 SQL injection

SQL Injection (SQLi) é um tipo de vulnerabilidade de injeção de código em que um invasor insere comandos ou consultas SQL maliciosas em entradas fornecidas a uma aplicação, com a intenção de fazer essas instruções executarem no banco de dados de forma não prevista ([OWASP Foundation, 2024b](#)). Essa falha ocorre tipicamente quando

a aplicação concatena diretamente dados de entrada (por exemplo, campos de formulário ou parâmetros de URL) dentro de consultas SQL, sem a devida sanitização ou validação, permitindo que um atacante modifique a estrutura da consulta original (HALFOND; VIEGAS; ORSO, 2006).

Uma exploração SQLi bem-sucedida pode resultar em acesso não autorizado a dados sensíveis, alteração ou deleção de registros, execução de comandos administrativos no Sistema Gerenciador de Banco de Dados (SGBD) (como desligar o servidor ou manipular permissões) e até, em certos casos, comprometer o sistema operacional hospedeiro através de extensões do banco de dados (OWASP Foundation, 2024b). Por exemplo, um atacante poderia inserir `' OR '1'='1` em um campo de login, levando a uma consulta SQL que sempre resulta verdadeira e efetivamente burlando a autenticação, obtendo acesso indevido ao sistema. A literatura aponta o SQLi como uma das vulnerabilidades mais graves e recorrentes em aplicações web nas últimas décadas (HALFOND; VIEGAS; ORSO, 2006).

As consequências de um SQLi bem explorado podem ser desastrosas, incluindo roubo de grandes volumes de dados (como listas de usuários e senhas), violação de integridade (dados corrompidos ou falsificados) e perda de controle sobre o banco de dados. Para mitigar SQLi, recomenda-se fortemente o uso de consultas preparadas (*prepared statements*) ou ORM com parametrização (evitando a montagem dinâmica de SQL), validação rigorosa de entradas, uso de mínimos privilégios para contas de banco de dados e configuração de *firewalls* de aplicação (WAF) com regras para detectar padrões de ataque (OWASP Foundation, 2024b). A prevenção eficaz de SQLi é essencial para assegurar que a aplicação mantenha a confidencialidade e a integridade de suas informações armazenadas.

2.4.3 *Server-side request forgery*

Server-Side Request Forgery (SSRF) é uma vulnerabilidade na qual o atacante induz um servidor vulnerável a fazer requisições a partir do próprio lado do servidor para recursos internos ou externos não intencionados, burlando restrições de acesso. Em um ataque SSRF, o invasor geralmente encontra uma funcionalidade na aplicação web que aceita uma URL ou endereço como entrada e consegue manipulá-la para que o servidor envie uma requisição para um destino inesperado (OWASP Foundation, 2024a). Como essas novas requisições partem do servidor, que muitas vezes tem acesso privilegiado à rede interna, o atacante pode explorar serviços internos que não seriam acessíveis externamente – por exemplo, consultar *endpoints* de API internas, ler metadados de instâncias em nuvem (como aqueles disponibilizados em `http://169.254.169.254` na AWS) contendo credenciais, ou interagir com bancos de dados e outros sistemas normalmente protegidos por *firewall* (OWASP Foundation, 2024a).

Em termos de impacto, o SSRF pode levar à exposição de informações sensíveis do ambiente interno e, em cenários avançados, encadear ataques para obtenção de acesso

mais profundo. Um caso real que ilustra a gravidade do SSRF foi o incidente de violação de dados da empresa Capital One, no qual a atacante explorou uma falha SSRF em uma aplicação hospedada na AWS para acessar chaves de credenciais da instância e, em seguida, extrair dados confidenciais de mais de 100 milhões de clientes do banco ([AL-TALAK; ABBASS, 2021](#)).

Nos últimos anos, a preocupação com SSRF aumentou substancialmente: a edição 2021 do OWASP Top 10 incluiu SSRF entre as dez principais categorias de risco (A10:2021), ressaltando que a severidade do SSRF tem crescido devido ao uso de serviços em nuvem e à complexidade das arquiteturas modernas ([AL-TALAK; ABBASS, 2021](#)). A mitigação de SSRF requer medidas como validação rigorosa e saneamento de URLs fornecidas pelo usuário, restrição do tráfego de saída do servidor a apenas endereços necessários, segmentação de redes internas e uso de listas de permissão de destinos externos ([AL-TALAK; ABBASS, 2021](#)).

2.5 Trabalho correlatos

A segurança em ambientes de computação em nuvem tem sido tema de diversos estudos que abordam desde ataques específicos até padrões de segurança e estudos de caso comparativos. A seguir, são apresentados três trabalhos que se alinham ao contexto deste estudo:

Neste artigo ([PATIL, 2024](#)) é proposta uma estratégia abrangente para mitigar ataques de negação de serviço distribuídos (DDoS) em ambientes AWS. O estudo combina técnicas de mascaramento de dados com os mecanismos nativos de proteção da AWS como AWS Shield, AWS WAF e Amazon CloudFront para detectar e bloquear tráfego malicioso antes que este cause uma sobrecarga e eventual indisponibilidade no sistema. Os autores enfatizam a necessidade de monitoramento contínuo e de uma arquitetura resiliente para enfrentar os desafios impostos por ataques DDoS.

Além disso, o artigo discute a integração dessas ferramentas com práticas recomendadas de segurança, como a implementação de políticas de controle de acesso e a utilização de *logs* para análise de tráfego. Essa abordagem integrada visa não apenas a mitigação de ataques, mas também a prevenção e a resposta rápida a incidentes de segurança.

Outro artigo relevante ao presente estudo é ([RATH et al., 2019](#)), que apresenta um conjunto de padrões de segurança voltados para aplicações na nuvem, abordando aspectos que vão desde a segurança do sistema e dos dados até a privacidade. O estudo fornece uma metodologia para definir padrões de segurança e os aplica em estudos de caso na AWS e no Azure.

O estudo fornece uma análise de soluções oferecidas pela AWS e Azure, mostrando

que soluções poderiam ser utilizadas para atender aos padrões de segurança definidos. Assim, este estudo se relaciona com o presente trabalho ao apresentar serviços da AWS que podem ajudar a tratar pontos importantes relacionados à segurança em nuvem.

Já em (SINGH, 2019) é feito um comparativo sobre segurança em computação em nuvem, focando nas plataformas AWS e Google Cloud. O artigo examina vulnerabilidades comuns, como configurações incorretas de permissões e falta de criptografia, e propõe soluções para mitigar esses riscos.

Além de destacar a importância de definir bem as políticas de segurança na nuvem, o estudo apresenta uma série de ferramentas nativas dos provedores de nuvem que podem ajudar no monitoramento e proteção dos dados das organizações. Essas soluções incluem mecanismos de auditoria, criptografia de dados e detecção de vulnerabilidades.

3 Desenvolvimento

Para o desenvolvimento deste trabalho, explora-se um estudo de caso sobre a proteção de servidores web na AWS, apresentando ataques comuns e as respectivas estratégias de mitigação. Cada seção aborda um tipo específico de ataque, explica como ele funciona, demonstra seu impacto em um ambiente sem proteções adequadas e, finalmente, detalha as medidas de segurança que podem ser implementadas na AWS para mitigar esses ataques.

3.1 Metodologia

Cada cenário estudado é dividido em várias seções para garantir uma abordagem mais detalhada de cada tópico. A seguir, são descritos os tópicos que compõem cada caso de uso:

- Descrição do cenário de testes
- Cenário de ataque
- Implementação das medidas de segurança
- Avaliação e análise dos resultados

3.1.1 Descrição do cenário de testes

Nesta seção, apresenta-se o ambiente de testes utilizado para o estudo de caso. Inclui a descrição dos recursos computacionais e de rede utilizados, e a configuração básica sem medidas de segurança adicionais. São fornecidos diagramas da arquitetura inicial, especificações técnicas das instâncias e justificativas para a escolha dos componentes do ambiente.

3.1.2 Cenário de ataque

Aqui são apresentados os cenários de ataque que serão simulados para testar a segurança do ambiente de nuvem. Isso inclui uma explicação dos tipos de ataques selecionados, bem como a metodologia utilizada para simular esses ataques de forma controlada. São descritos os objetivos de cada ataque e os potenciais impactos no ambiente desprotegido.

3.1.3 Implementação das medidas de segurança

Nesse tópico, são descritas as medidas de segurança implementadas para mitigar os ataques simulados, detalhando os passos para configuração e integração dessas ferramentas no ambiente de testes. São incluídos diagramas atualizados da arquitetura com as proteções implementadas, além de explicações sobre como cada medida contribui para a mitigação dos ataques simulados.

3.1.4 Avaliação e análise dos resultados

Na última parte são apresentados os resultados da implementação das medidas de segurança. É feita uma análise comparativa entre o impacto dos ataques no ambiente desprotegido e no ambiente protegido, destacando os benefícios observados e a eficácia das soluções implementadas. São discutidas eventuais limitações encontradas e recomendações para aprimoramentos futuros. A análise será suportada por gráficos e/ou tabelas que ilustrarão os dados coletados durante os testes, proporcionando uma visão clara dos efeitos das medidas de segurança aplicadas.

3.2 Ataque de negação de serviço - DoS

3.2.1 Descrição do cenário de testes

Para a realização dos testes de negação de serviço, foi utilizada a arquitetura na AWS como na Figura 1. Na parte do servidor, uma instância EC2 do tipo t3.micro foi configurada, com IP público e acesso à internet. A máquina possui 1GB de memória RAM e 1 VCPU, com uma largura de banda de até 5 Gbps. Foram instalados Apache2, PHP e MySQL para o funcionamento do site.

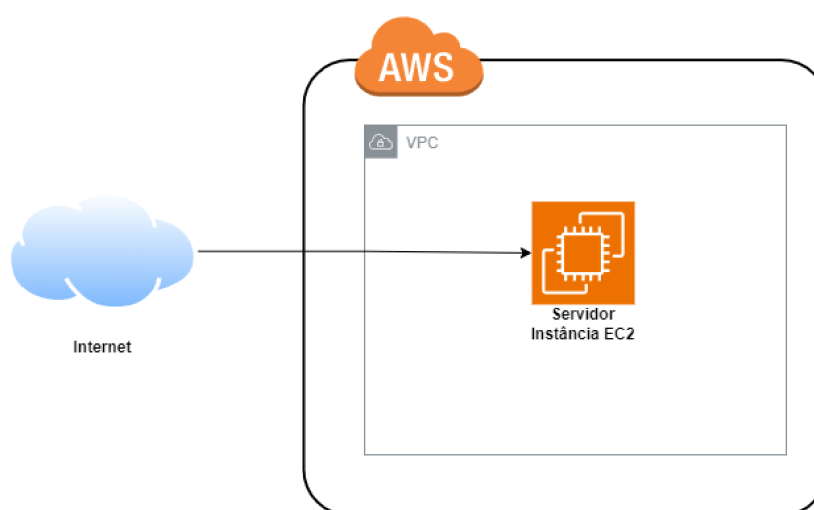


Figura 1 – Diagrama da arquitetura inicial dos testes na AWS. Fonte: Do Autor.

A aplicação hospedada consiste em uma página de controle de estoque de uma loja virtual simples, desenvolvida pelo próprio autor e disponível em <https://github.com/resendetarcisio/tcc-cloud/blob/main/use-case-dos-index.php>. Esse sistema exibe quantas unidades estão disponíveis de cada item, seu preço e descrição, como mostra a Figura 2. A interface foi implementada em PHP e comunica-se com o banco de dados MySQL para a persistência dos dados.

Produtos Disponíveis				
ID	Nome	Descrição	Preço (R\$)	Quantidade
1	Camiseta Básica	Camiseta de algodão 100% com diversas cores.	29,90	50
2	Calça Jeans	Calça jeans masculina com modelagem slim.	89,90	30
3	Tênis Esportivo	Tênis para corrida com solado reforçado.	149,90	20
4	Relógio de Pulso	Relógio de pulso resistente à água.	199,90	15
5	Mochila Escolar	Mochila escolar com vários compartimentos.	79,90	25
6	Cadeira Gamer	Cadeira ergonômica para jogos com ajuste de altura e inclinação.	599,90	10
7	Smartphone 64GB	Smartphone com tela de 6.5 polegadas e 64GB de armazenamento.	1.299,90	20
8	Fone de Ouvido Bluetooth	Fone de ouvido sem fio com cancelamento de ruído.	249,90	40
9	Teclado Mecânico	Teclado mecânico com retroiluminação RGB.	329,90	15
10	Mouse Gamer	Mouse óptico com 16000 DPI e iluminação RGB.	199,90	25
11	Monitor 24 Polegadas	Monitor Full HD de 24 polegadas com bordas finas.	899,90	12
12	Impressora Multifuncional	Impressora multifuncional com scanner e Wi-Fi.	499,90	8
13	Tablet 10 Polegadas	Tablet com tela de 10 polegadas e 32GB de armazenamento.	999,90	18
14	HD Externo 1TB	HD externo portátil de 1TB com conexão USB 3.0.	349,90	30
15	Carregador Portátil 10000mAh	Carregador portátil com capacidade de 10000mAh e duas saídas USB.	129,90	50

Figura 2 – Página do site. Fonte: Do Autor.

3.2.2 Cenário de ataque

No cenário de ataque de *Denial of Service (DoS)*, o objetivo foi gerar o maior volume possível de requisições para sobrecarregar a instância EC2 e, eventualmente, impedir que ela respondesse a requisições legítimas. Para isso, desenvolveu-se um *script* em Python, disponível em <https://github.com/resendetarcisio/tcc-cloud/blob/main/use-case-dos-stress-test.py>, que envia centenas de requisições HTTP por segundo à página web de estoque. O código abusa de conexões simultâneas e repete chamadas de forma contínua, buscando exaurir CPU e a memória da instância.

Durante os testes com a execução do *script*, foi atingida uma taxa máxima de 166 requisições por segundo, o que elevou o uso de CPU significativamente e saturou a fila de processos do Apache. Os gráficos de monitoramento de uso da instância da Figura 3 e a listagem de processos da Figura 4 confirmaram a sobrecarga do sistema.

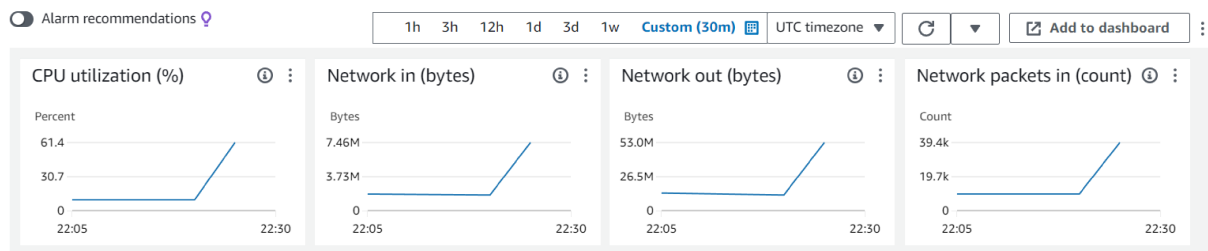


Figura 3 – Gráfico de utilização da instância na AWS. Fonte: Do Autor.

```
top - 21:29:43 up 4 min, 1 user, load average: 61.00, 30.97, 12.48
Tasks: 223 total, 44 running, 179 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6.8 us, 2.6 sy, 0.0 ni, 0.2 id, 0.0 wa, 0.0 hi, 1.0 si, 89.5 st
MiB Mem : 914.2 total, 92.3 free, 848.5 used, 107.0 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used, 65.7 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
757	mysql	20	0	1873856	431412	12160	R	105.1	46.1	2:59.41	mysqld
1166	www-data	20	0	207664	15228	9472	R	4.8	1.6	0:01.80	apache2
1268	www-data	20	0	207664	15364	9600	S	4.5	1.6	0:01.71	apache2
1778	www-data	20	0	207664	15228	9472	S	4.5	1.6	0:02.44	apache2
1976	www-data	20	0	207664	15252	9600	S	4.5	1.6	0:02.34	apache2
1750	www-data	20	0	207664	15356	9600	R	4.2	1.6	0:02.02	apache2
1531	www-data	20	0	207664	15356	9600	S	2.9	1.6	0:02.21	apache2
1073	www-data	20	0	207664	15356	9600	S	2.6	1.6	0:01.81	apache2
1136	www-data	20	0	207664	15228	9472	R	2.6	1.6	0:01.43	apache2
1170	www-data	20	0	207664	15228	9472	S	2.6	1.6	0:02.14	apache2
1239	www-data	20	0	207664	15356	9600	R	2.6	1.6	0:02.10	apache2
1266	www-data	20	0	207664	15228	9472	S	2.6	1.6	0:01.34	apache2
1512	www-data	20	0	207664	15356	9600	R	2.6	1.6	0:01.83	apache2
1517	www-data	20	0	207664	15356	9600	R	2.6	1.6	0:01.74	apache2
1520	www-data	20	0	207664	15356	9600	R	2.6	1.6	0:01.68	apache2
1527	www-data	20	0	207664	15356	9600	S	2.6	1.6	0:01.72	apache2
1751	www-data	20	0	207664	15484	9728	R	2.6	1.7	0:01.41	apache2
611	www-data	20	0	207920	15484	9728	R	2.2	1.7	0:02.33	apache2
1069	www-data	20	0	207664	15356	9600	R	2.2	1.6	0:01.76	apache2
1088	www-data	20	0	207664	15356	9600	R	2.2	1.6	0:01.75	apache2

Figura 4 – Listagem de processos e utilização de recursos. Fonte: Do Autor.

Quando a taxa de requisições foi aumentada além de 166 requisições por segundo, a instância passou a rejeitar conexões e, após alguns segundos, parou de responder completamente. Os navegadores retornaram erro de conexão expirada e as chamadas `curl` exibiram `timeout`, como mostrado na Figura 5 e na Figura 6 respectivamente.

A conexão expirou

O servidor  demorou muito para responder.

- Este site pode estar temporariamente indisponível ou sobrecarregado. Tente novamente daqui a pouco.
- Se você não conseguir carregar nenhuma página, verifique a conexão de rede do computador.
- Se a rede ou o computador estiver protegido por um firewall ou proxy, verifique se o Firefox está autorizado a acessar a web.

[Tentar novamente](#)

Figura 5 – Página fora do ar. Fonte: Do Autor.

```
$ curl --max-time 10 http:// /index.php
curl: (28) Operation timed out after 10000 milliseconds with 0 bytes received
$
```

Figura 6 – Curl para o site recebe *timeout*. Fonte: Do Autor.

3.2.3 Implementação das medidas de segurança

Após confirmar a vulnerabilidade da aplicação ao ataque DoS, foram implementadas contramedidas. Para mitigar esse tipo de ataque, foi utilizado o *AWS Web Application Firewall (WAF)*, um serviço gerenciado de *firewall* para aplicações web que monitora e filtra o tráfego HTTP/HTTPS com base nas regras definidas, em conjunto com um *Application Load Balancer (ALB)*, um balanceador de carga de camada de aplicação que distribui requisições HTTP/HTTPS entre instâncias e faz a terminação TLS, permitindo a integração com o WAF. A arquitetura de segurança proposta é ilustrada na Figura 7:

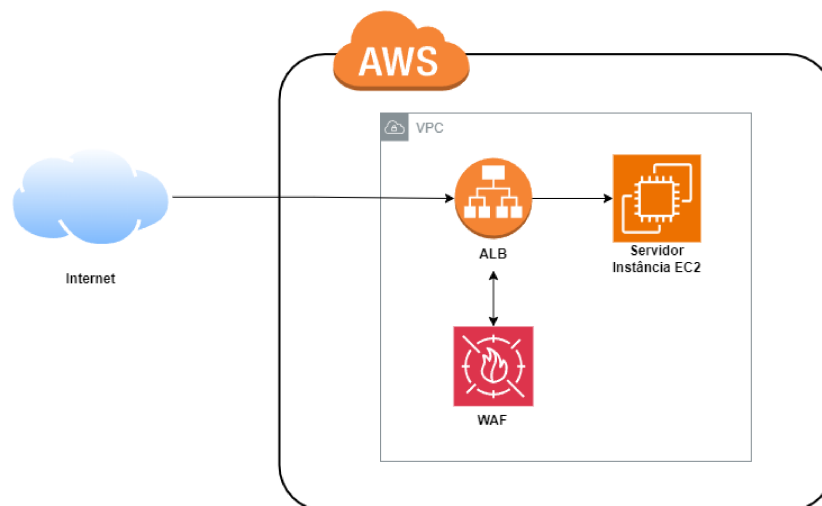


Figura 7 – Diagrama da arquitetura proposta com ALB e WAF. Fonte: Do Autor.

Na solução proposta, o WAF foi integrado ao ALB, que atua como ponto de entrada para a aplicação. Embora o balanceador de carga não esteja sendo usado para distribuir o tráfego entre múltiplas instâncias, ele permite a terminação TLS e a aplicação de regras de segurança no WAF, algo que não seria possível diretamente na instância EC2.

Com o WAF, criaram-se regras específicas para mitigar o ataque DoS. Uma delas foi a regra de *Rate Limit*, configurada para bloquear IPs que excedam 100 requisições por minuto, como mostrado na Figura 8:

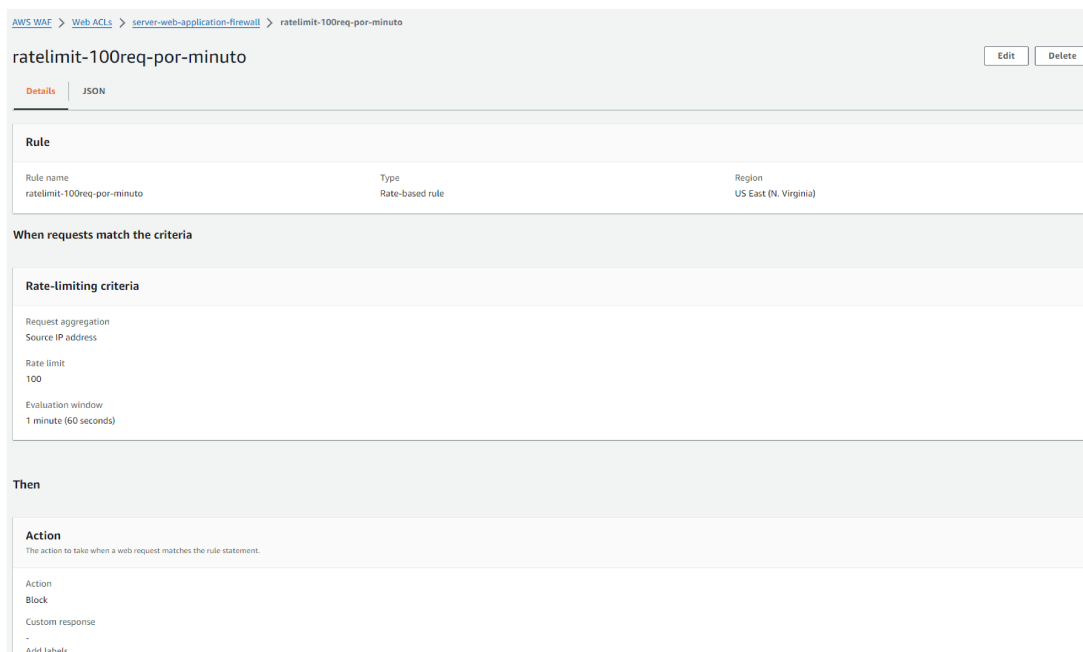


Figura 8 – Regra de *Rate Limit* aplicada no WAF. Fonte: Do Autor.

Durante os testes subsequentes, o WAF bloqueou temporariamente todo IP que ultrapassasse 100 chamadas por minuto. A Figura 9 mostra o bloqueio de uma requisição executada com *curl*, após o endereço IP ter feito uma quantidade maior de requisições do que o permitido na configuração da regra.

```
c:\>curl -X HEAD -I http://[redacted].us-east-1.elb.amazonaws.com/index.php
HTTP/1.1 403 Forbidden
Server: awselb/2.0
Date: Sun, 01 Sep 2024 23:16:52 GMT
Content-Type: text/html
Content-Length: 118
Connection: keep-alive
```

Figura 9 – Requisição para o site bloqueada. Fonte: Do Autor.

Para aumentar a flexibilidade, testou-se também a implementação de *CAPTCHA* em vez de bloqueio imediato. Nos testes, quando um endereço IP extrapolava o número de requisições, o cliente era solicitado a resolver um desafio de *CAPTCHA*, como ilustrado na Figura 10

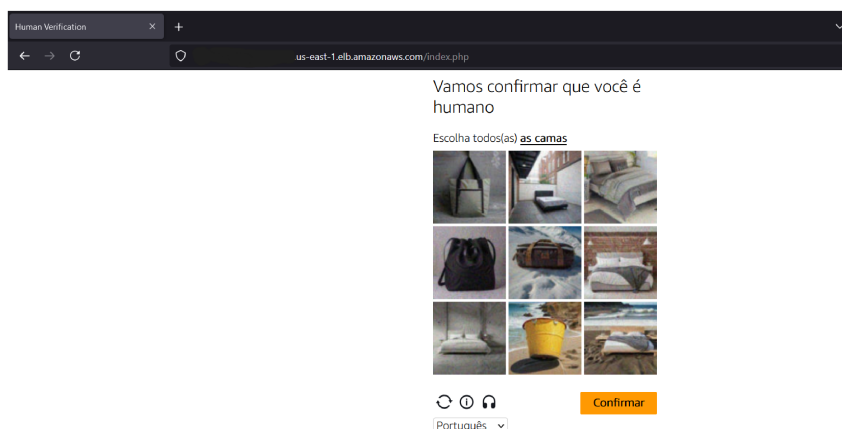


Figura 10 – *Captcha* mostrado após atingir limite de requisições. Fonte: Do Autor.

Esse mecanismo permite distinguir humanos de *bots*: apenas *scripts* automatizados permaneciam bloqueados, enquanto usuários verdadeiros, ao resolver o *CAPTCHA*, recuperam o acesso sem afetar a navegação. Com esta abordagem, mesmo que uma mesma origem IP envie muitas requisições ao site, a utilização do site por usuários legítimos não é inviabilizada.

3.2.4 Avaliação e análise dos resultados

As medidas implementadas provaram-se eficazes na mitigação de ataques DoS de uma única origem. Ao configurar um limite de requisições, o AWS WAF conseguiu bloquear tentativas de inundar o servidor com tráfego excessivo. Tanto o bloqueio por IP quanto o uso de *CAPTCHA* impediram que o ataque comprometesse a disponibilidade do sistema.

Essas soluções oferecem proteção adequada para cenários em que o tráfego excessivo vem de uma única origem IP. Contudo, é importante observar que, em um ataque DDoS (*Distributed Denial of Service*), em que as requisições provêm de múltiplos IPs distribuídos, o WAF configurado desta forma pode não ser tão eficaz, uma vez que o limite por IP não é suficiente para detectar e bloquear o tráfego malicioso. Para lidar com DDoS, é necessário implementar soluções adicionais, como escalar a capacidade computacional e também *AWS Shield*, que oferece proteção mais robusta contra ataques distribuídos.

Outro aspecto importante é a personalização das regras de segurança. A configuração de um limite de requisições por IP pode não ser trivial e pode variar conforme o tráfego esperado pela aplicação. Uma análise detalhada do comportamento típico dos usuários deve ser conduzida para definir limites apropriados. Soluções como *CAPTCHA* também oferecem um equilíbrio entre segurança e experiência do usuário, ao evitar bloqueios automáticos em situações em que há alta demanda legítima.

3.3 *SQL injection - SQLi*

3.3.1 Descrição do cenário de testes

Para os testes de *SQL Injection*, foi utilizada a arquitetura na AWS ilustrada na Figura 11. Essa configuração permitiu avaliar o comportamento da aplicação e as respostas do sistema sob tentativas de injeção de comandos SQL.

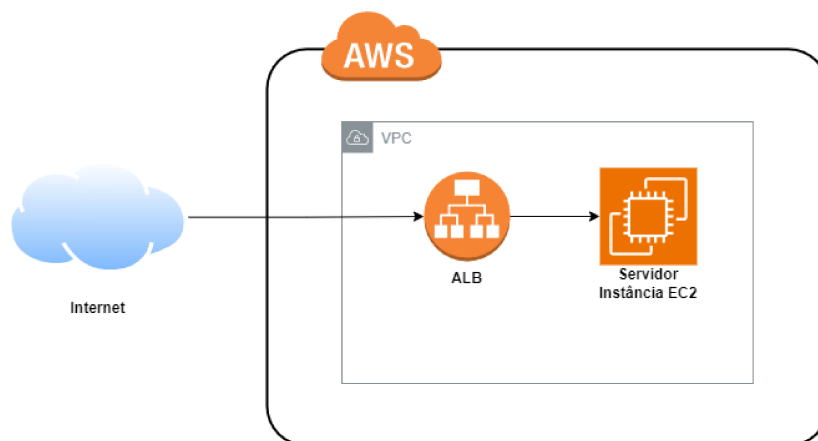


Figura 11 – Arquitetura de testes para *SQL Injection*. Fonte: Do Autor.

Foi configurada uma instância EC2 do tipo t3.micro, com 1GB de memória RAM, 1 vCPU e largura de banda de até 5 Gbps. A instância foi configurada sem IP público, e um *Application Load Balancer* foi adicionado para encaminhar o tráfego público à aplicação de forma controlada.

O servidor foi configurado com Apache2, PHP e MySQL. Foi utilizado o mesmo sistema de controle de estoque desenvolvido pelo autor, agora complementado por uma página de autenticação simples (disponível em <<https://github.com/resendetarcisio/tcc-cloud/blob/main/use-case-sqli-login.php>>), ilustrada na Figura 12, que restringe o acesso à página de produtos apenas a usuários autenticados. No entanto, essa página não implementa mecanismos de sanitização ou validação de entradas, tornando-a suscetível a injeções SQL.

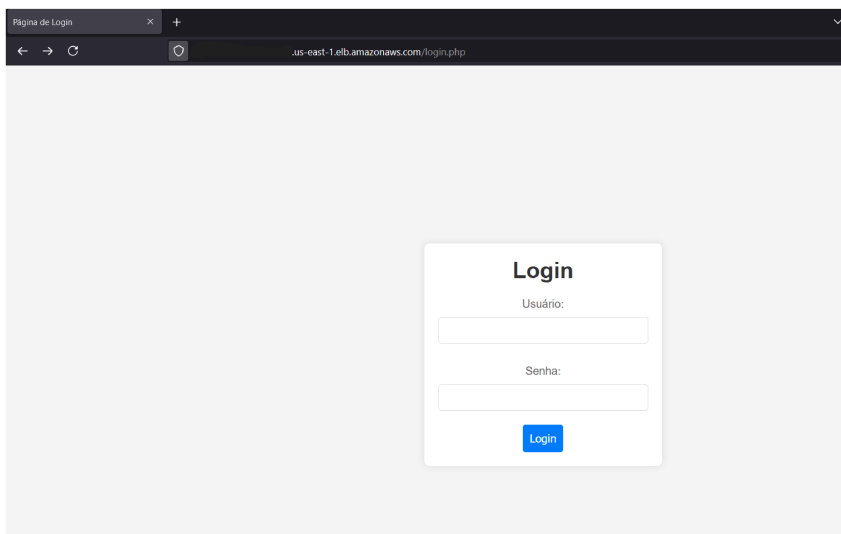


Figura 12 – Página de login. Fonte: Do Autor.

3.3.2 Cenário de ataque

O objetivo do ataque era autenticar no site sem conhecer o login e a senha de um usuário válido, explorando a falta de validação dos parâmetros de entrada. Para isso, utilizou-se um *payload* — um trecho de dados maliciosos enviado à aplicação para explorar vulnerabilidades — capaz de forçar a cláusula **WHERE** a retornar verdadeiro para qualquer credencial informada.

Durante o teste, foi inserido o seguinte código no campo de login e senha: `' OR '1'='1'`. Esse *payload* é bastante utilizado em SQLi pois faz com que a condição lógica sempre seja satisfeita.

Dessa forma, a consulta SQL original:

```
SELECT * FROM usuarios WHERE usuario = '$usuario_formulario'
AND senha = '$senha_formulario'
```

foi transformada na seguinte instrução:

```
SELECT * FROM usuarios WHERE usuario = '' OR '1'='1'
AND senha = '' OR '1'='1'
```

A modificação garante que a expressão na cláusula **WHERE** seja verdadeira em todas as situações, anulando o controle de autenticação.

Como `'1'='1'` é uma expressão sempre verdadeira, a consulta retorna uma linha válida, permitindo a autenticação no sistema mesmo sem o conhecimento do login e senha

corretos. O sucesso do login é confirmado pela interface da página de login mostrando a frase "Login realizado com sucesso" e mostrando o botão de ir para a página da aplicação, conforme ilustrado na Figura 13. Com isso, foi possível concluir que a aplicação é vulnerável a ataques de *SQL Injection*.

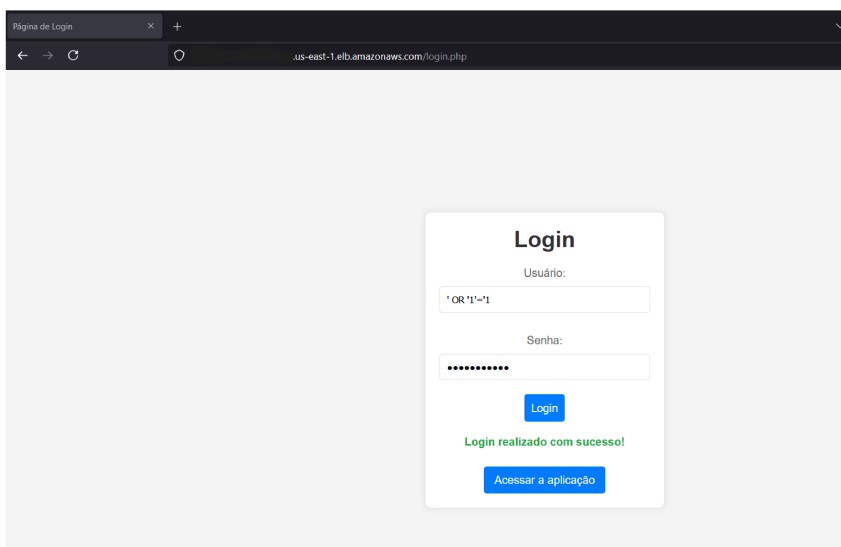


Figura 13 – Login realizado com sucesso. Fonte: Do Autor.

3.3.3 Implementação das medidas de segurança

Uma das principais medidas de segurança disponíveis na AWS para mitigar ataques de *SQL Injection* é a implementação de regras gerenciadas no AWS WAF. Essas regras são mantidas e atualizadas continuamente pela AWS e conseguem detectar padrões típicos de ataques de injeção SQL em requisições de usuários. Uma dessas regras é o conjunto *AWSManagedRulesSQLiRuleSet*, detalhado na Figura 14.

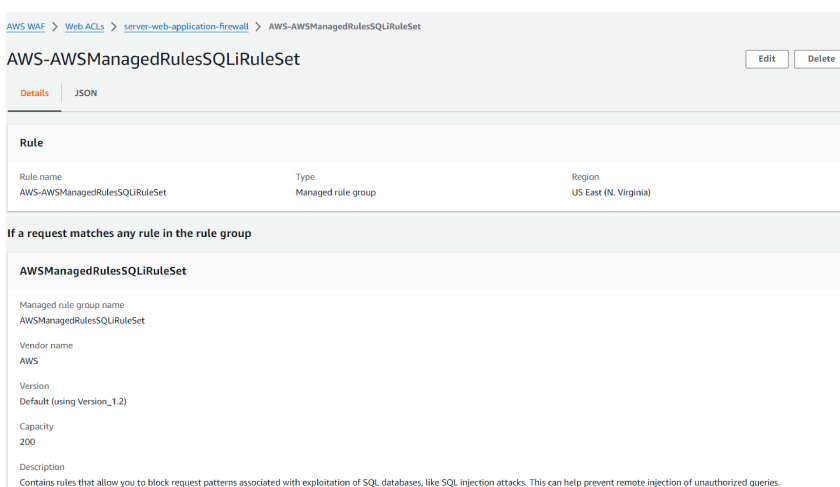


Figura 14 – Detalhes do grupo de regras *AWSManagedRulesSQLiRuleSet*. Fonte: Do Autor.

Para aplicar a proteção, o WAF foi integrado ao *Load Balancer* e o conjunto *AWSManagedRulesSQLiRuleSet* foi habilitado para interceptar e bloquear tentativas de exploração. Essa configuração direciona todo o tráfego do site ao WAF antes de chegar ao servidor de aplicação.

Após a aplicação da regra, o mesmo ataque foi executado, mas a requisição foi bloqueada, conforme demonstra a Figura 15. O bloqueio ocorreu no nível do WAF, prevenindo qualquer execução subsequente no banco de dados MySQL.



Figura 15 – Requisição maliciosa bloqueada pelo WAF. Fonte: Do Autor.

Ao analisar os *logs* do WAF, confirmou-se que o WAF detectou um padrão típico de *SQL Injection* no corpo da requisição. O registro de log na Figura 16 evidencia os detalhes da detecção e bloqueio pelo grupo de regras configurado anteriormente.

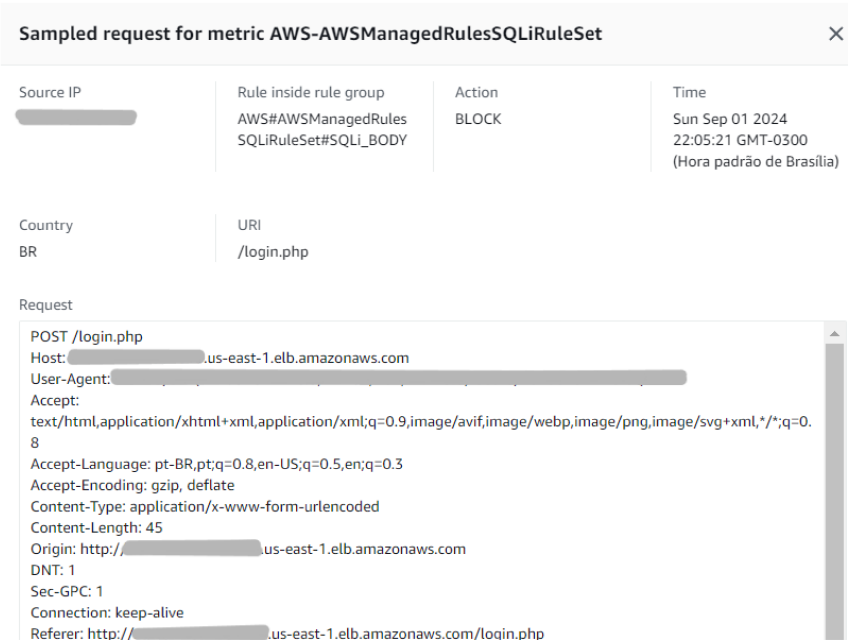


Figura 16 – Log com requisição maliciosa detectada e bloqueada. Fonte: Do Autor.

3.3.4 Avaliação e análise dos resultados

A implementação do AWS WAF com a regra *AWSManagedRulesSQLiRuleSet* se mostrou eficaz na detecção e bloqueio de tentativas de *SQL Injection*. A solução ofereceu uma camada adicional de proteção, útil para aplicações que utilizam sistemas SQL e recebem entradas de usuários.

No entanto, é fundamental destacar que essa abordagem é uma medida paliativa e não resolve diretamente a vulnerabilidade de *SQL Injection*. Nesse caso, o WAF vai dificultar a exploração, mas não eliminar a possibilidade de ataques, especialmente se técnicas mais avançadas forem utilizadas para tentar contornar as detecções e bloqueios. Portanto, a correção definitiva envolveria ajustes na forma como a consulta SQL é preparada e validação adequada de entradas fornecidas pelos usuários.

Embora o WAF ajude a mitigar ataques, ele deve ser visto como uma camada adicional de segurança e não como uma solução definitiva. A eliminação da vulnerabilidade de *SQL Injection* depende de práticas seguras de codificação da aplicação.

3.4 *Server-side request forgery* - SSRF

3.4.1 Descrição do cenário de testes

Para o cenário de teste da vulnerabilidade *Server-Side Request Forgery (SSRF)*, foi utilizada uma arquitetura semelhante à empregada nos testes de *SQL Injection*. Utilizou-se uma instância EC2 do tipo t3.micro, com 1 GB de memória RAM, 1 vCPU e IP público para acesso à internet. Adicionalmente, a opção *Instance Metadata Service version 1 (IMDSv1)* foi mantida habilitada na configuração da instância.

A aplicação, desenvolvida pelo próprio autor e disponível em <https://github.com/resendetarcisio/tcc-cloud/blob/main/use-case-ssrf-resumo-site.php>, é ilustrada na Figura 17. Trata-se de uma página web que recebe uma URL informada pelo usuário, faz uma requisição HTTP ao endereço fornecido e, com base no conteúdo retornado, utiliza a *API* do ChatGPT para gerar um resumo automático da página. Esse fluxo de requisição-resposta demonstra como a funcionalidade de sumarização pode ser explorada para SSRF quando não há validação adequada da entrada.

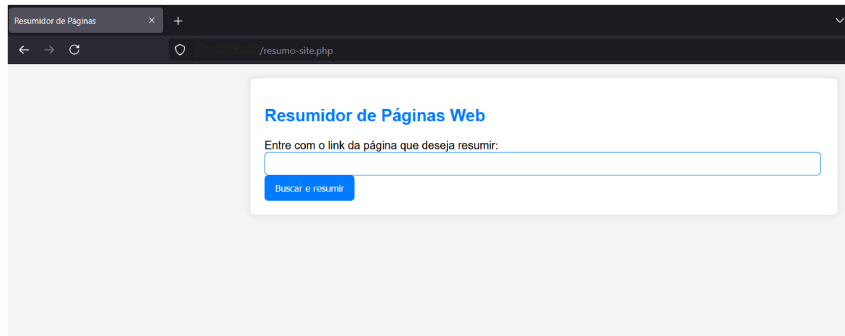


Figura 17 – Site que resume páginas dado uma URL. Fonte: Do Autor.

3.4.2 Cenário de ataque

No cenário de ataque por SSRF, o objetivo principal é a obtenção das credenciais da instância, por meio do serviço de metadados de instância. Esse serviço, normalmente acessível apenas internamente pela própria instância, fornece informações sensíveis e operacionais, como a região da instância, IP, nome da Role associada e as respectivas credenciais temporárias — sendo este último o foco deste experimento.

A posse dessas credenciais pode permitir que a instância acesse outros serviços da AWS, como armazenamento, banco de dados e demais instâncias, com os privilégios atribuídos à sua *Role*. Assim, o ataque visa induzir o servidor a realizar uma requisição, sob comando do atacante, ao serviço de metadados, retornando como resposta as informações desejadas.

Inicialmente, observa-se que o site aceita uma URL informada pelo usuário e realiza uma requisição ao conteúdo correspondente, com o intuito de gerar o resumo. A Figura 18 mostra a aplicação resumindo uma notícia qualquer de uma URL fornecida pelo usuário.

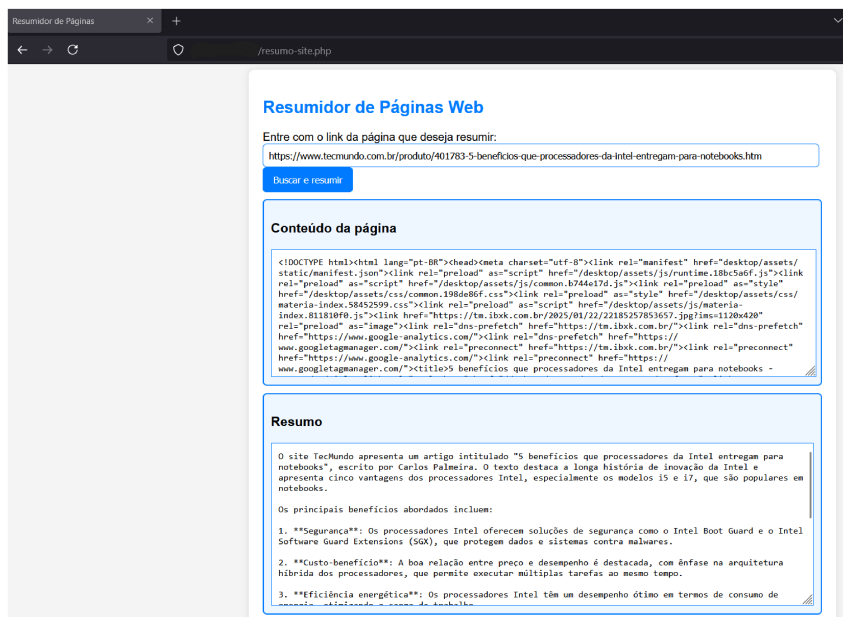


Figura 18 – Exemplo do funcionamento do site. Fonte: Do Autor.

Dessa forma, é possível verificar se a aplicação aceita qualquer URL arbitrária. Para isso, utilizou-se o IP reservado do serviço de metadados da AWS: 169.254.169.254.

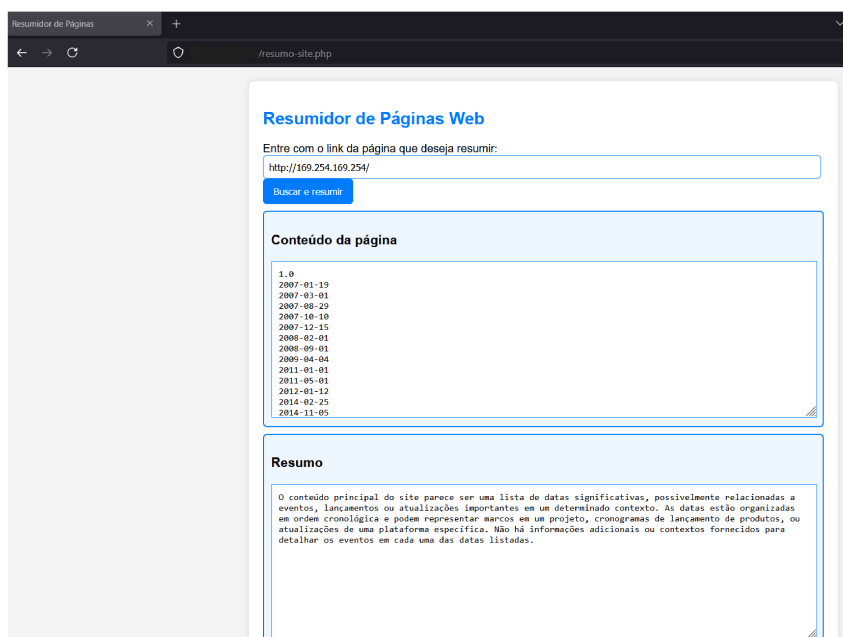


Figura 19 – Testando consultar o endereço do serviço de metadados. Fonte: Do Autor.

Como apresentado na Figura 19, o servidor enviou a requisição ao endereço de metadados especificado. Como a instância possui permissão para acessar o serviço de metadados da AWS, foi recebida uma resposta válida contendo as informações solicitadas. Em seguida, a aplicação exibiu o conteúdo retornado diretamente na interface, viabilizando sua visualização.

A partir deste ponto, consultou-se o caminho `/latest/meta-data/iam/security-credentials`, o qual retorna o nome da *Role* associada à instância. O resultado da consulta pode ser visto na Figura 20.

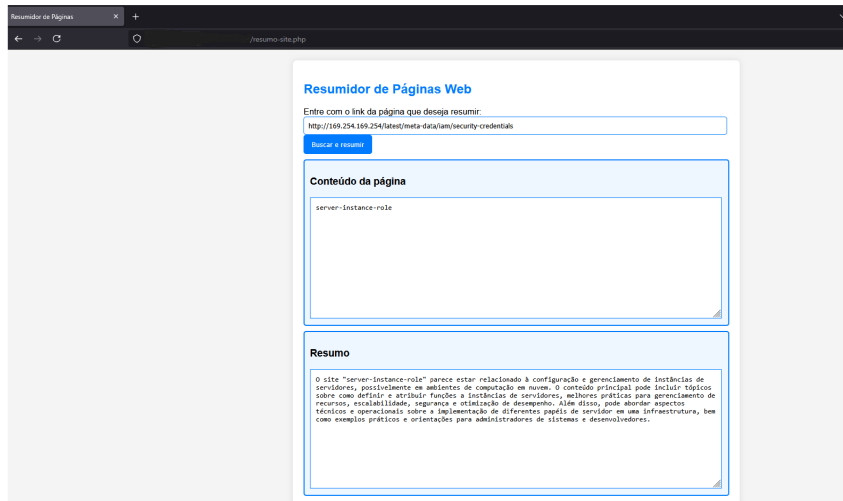


Figura 20 – Consultando o nome da *Role* da instância servidor. Fonte: Do Autor.

A resposta revelou que a instância possui uma *Role* denominada *server-instance-role*. De posse dessa informação, foi possível consultar o caminho completo: `/latest/meta-data/iam/security-credentials/server-instance-role`. O resultado é apresentado na Figura 21.

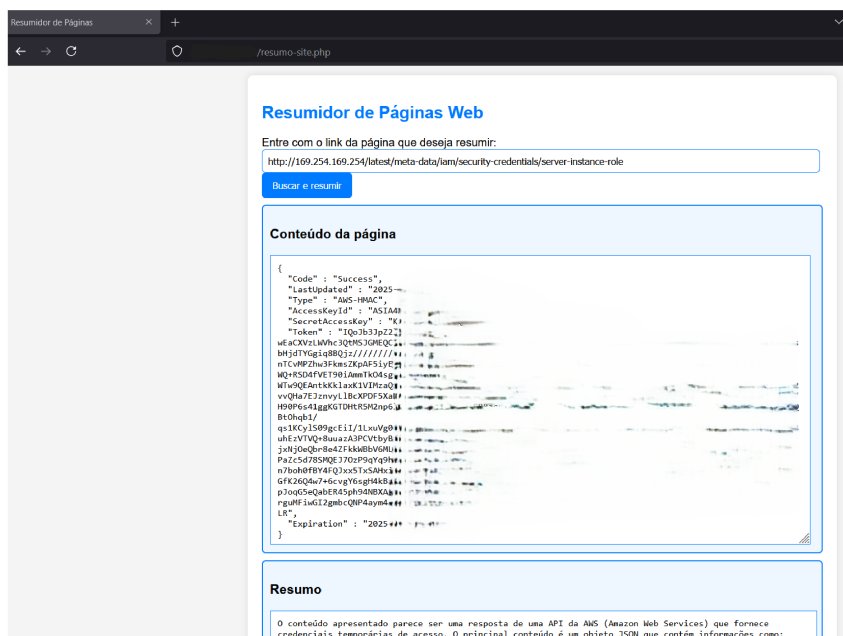


Figura 21 – Obtendo credenciais temporárias da *Role*. Fonte: Do Autor.

Os campos *AccessKeyId*, *SecretAccessKey* e *Token* contêm todas as credenciais necessárias para realizar chamadas autenticadas aos serviços da AWS, com os mesmos privilégios atribuídos à *Role* da instância comprometida.

3.4.3 Implementação das medidas de segurança

Como demonstrado anteriormente, a aplicação apresentou vulnerabilidade à técnica de SSRF, permitindo a extração de dados sensíveis por meio do serviço de metadados da instância. A seguir, são descritas medidas de segurança que podem ser implementadas na AWS para mitigar ou evitar esse tipo de exploração.

A primeira medida consiste em revisar a configuração do serviço de metadados da instância, desabilitando-o caso não seja necessário ou forçando a utilização exclusiva da versão IMDSv2, como mostra a Figura 22. Essa versão aprimorada requer a geração e o envio de um *token* de autenticação no cabeçalho da requisição, o que dificulta ataques como o apresentado, visto que, na aplicação vulnerável, não é possível controlar os *headers* enviados.

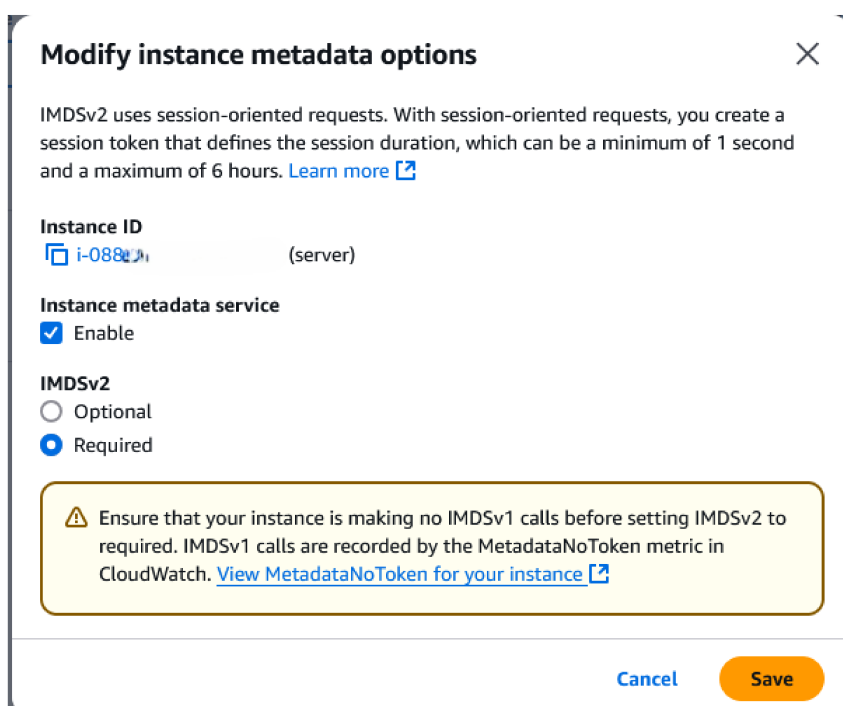


Figura 22 – Aplicando configuração de IMDSv2. Fonte: Do Autor.

Com essa configuração aplicada, a obtenção de credenciais por meio de uma simples requisição deixou de ser viável. E como mostra a Figura 23, o mesmo ataque falha ao tentar obter as credenciais.

Resumidor de Páginas Web

Entre com o link da página que deseja resumir:

<http://169.254.169.254/latest/meta-data/iam/security-credentials>

Buscar e resumir

Conteúdo da página

Error: Unable to fetch the page content.

Resumo

N/A

Figura 23 – Consultando credenciais após configurar IMDSv2. Fonte: Do Autor.

Adicionalmente, outra medida eficiente é a implementação de um *Web Application Firewall (WAF)*, configurado com o conjunto de regras gerenciado *AWSManagedRules-CommonRuleSet*. A arquitetura com a proteção implementada é ilustrada na Figura 24.

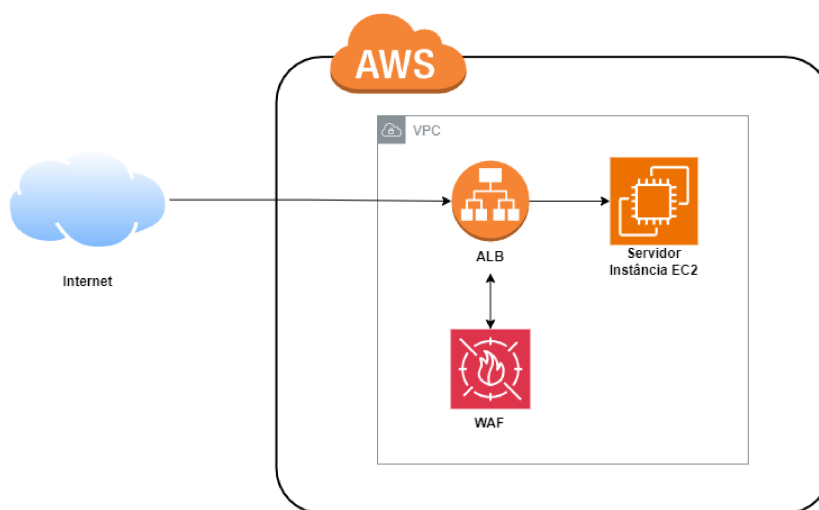


Figura 24 – Arquitetura com WAF aplicado. Fonte: Do Autor.

Após a aplicação do WAF e das regras mencionadas, novos testes foram realizados

e o ataque foi devidamente bloqueado. Nos registros do WAF na Figura 25, foi possível observar que a tentativa de ataque foi detectada pela regra *EC2MetaDataSSRF_BODY*, pertencente ao grupo de regras *AWSManagedRulesCommonRuleSet*. Além do bloqueio ao IP do serviço de metadados, o WAF também identificou e bloqueou tentativas do mesmo ataque utilizando *localhost* e endereços internos da rede.

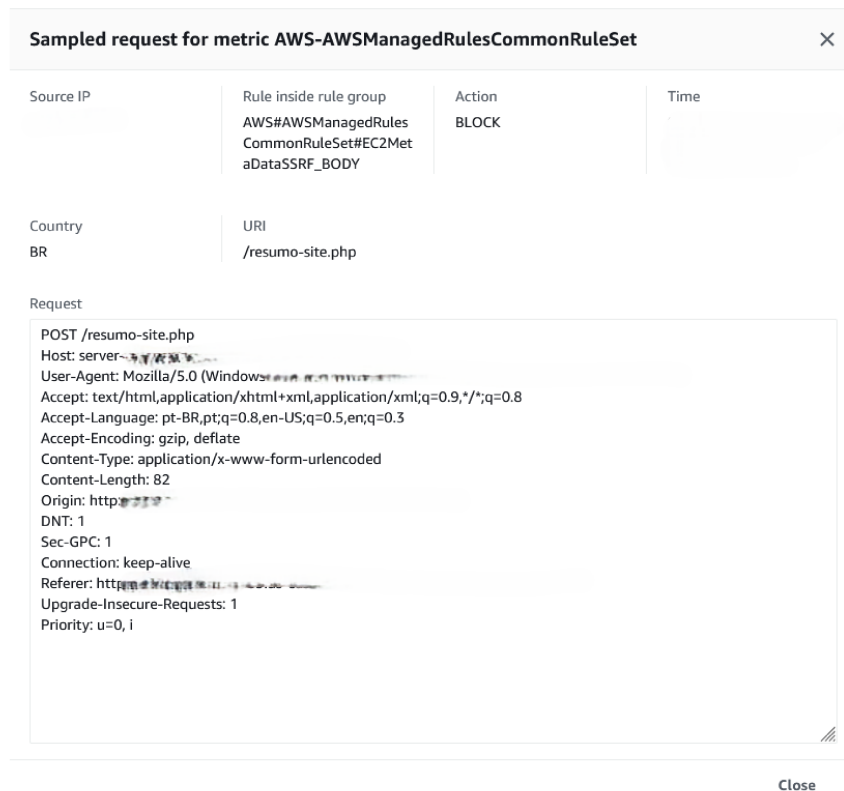


Figura 25 – Log de bloqueio do WAF, detectando SSRF. Fonte: Do Autor.

3.4.4 Avaliação e análise dos resultados

As medidas de mitigação implementadas apresentam diferentes formas de atuação. A obrigatoriedade do uso do IMDSv2 mostrou-se eficaz ao impedir a obtenção direta dos metadados por meio de requisições simples. Essa abordagem exige um ataque mais sofisticado, que seja capaz de gerar e utilizar um *token* de acesso. Contudo, essa proteção se limita à interface de metadados da instância, não prevenindo outros possíveis usos maliciosos da vulnerabilidade SSRF, como o acesso a serviços internos da rede ou portas não expostas externamente.

Por outro lado, a utilização do WAF com regras gerenciadas atua diretamente na camada de aplicação, possibilitando a detecção e o bloqueio de uma variedade maior de padrões maliciosos, incluindo tentativas de acesso a metadados e recursos internos da rede. Durante os testes, o WAF bloqueou não apenas as requisições aos metadados, como também aquelas direcionadas a outros recursos locais.

É importante destacar que a medida ideal consiste na correção da vulnerabilidade diretamente na aplicação, restringindo o acesso a URLs externas e implementando mecanismos de validação e sanitização de entradas. No entanto, as configurações mais seguras da instância (IMDSv2) e a utilização de regras no WAF contribuem significativamente para dificultar a exploração de SSRF em casos em que a vulnerabilidade ainda esteja presente.

Por fim, recomenda-se seguir a boa prática de conceder à *Role* da instância apenas os privilégios mínimos necessários. Dessa forma, caso as credenciais sejam comprometidas, o impacto sobre a infraestrutura será reduzido.

4 Conclusão

Ao longo deste trabalho foi possível observar, de forma prática, como ataques de negação de serviço (DoS), injeção de SQL (SQLi) e *Server-Side Request Forgery* (SSRF) exploram fragilidades comuns em aplicações web hospedadas na nuvem. Cada cenário de ataque demonstrou que ambientes sem proteções dedicadas ficam rapidamente sobrecarregados ou comprometidos, seja indisponibilizando o serviço, acessando dados não autorizados ou extraindo credenciais sensíveis.

A adoção de mecanismos nativos da AWS mostrou-se eficaz para mitigar esses riscos. No caso de DoS, o *rate limiting* e o uso de *CAPTCHA* no AWS WAF impediram que uma única origem IP sobrecarregasse os recursos da instância. Para SQLi, as regras gerenciadas do WAF barraram tentativas de injeção SQL. Já em SSRF, a combinação de IMDSv2 e as regras gerenciadas pela AWS no WAF bloqueou requisições ao serviço de metadados, protegendo as credenciais.

As defesas em nuvem devem ser complementadas por boas práticas de desenvolvimento. A validação adequada de entradas, o uso de consultas parametrizadas e uma visão crítica sobre as funcionalidades implementadas na aplicação são fundamentais para evitar a introdução de vulnerabilidades. Além disso, a configuração de privilégios mínimos nas *roles* é essencial para que, mesmo em caso de comprometimento, o impacto seja reduzido.

Por fim, este estudo reforça que a nuvem oferece poderosas ferramentas de segurança, mas é preciso saber configurá-las de forma correta, já que as proteções não são automaticamente habilitadas. Para isso, é imprescindível que as organizações adotem processos de revisão contínua de segurança, garantindo que as configurações estejam sempre alinhadas às melhores práticas. Somente assim é possível extrair todo o potencial dos recursos de segurança em nuvem e manter a confiabilidade dos serviços diante das ameaças em constante evolução.

Vale ressaltar, ainda, que este trabalho pode servir como um guia para estudantes e profissionais que desejam compreender o impacto de ataques em ambientes em nuvem e aprender a construir medidas de mitigação simples e diretas. No entanto, é importante destacar que existem algumas limitações deste estudo, dentre elas destaca-se o fato de que as soluções discutidas se aplicam exclusivamente à AWS, e podem não ser diretamente aplicáveis a outros provedores de nuvem, que possuem ferramentas e configurações específicas. Além disso, a implementação de recursos de segurança, como o WAF e o *Load Balancer*, pode acarretar custos adicionais significativos para a infraestrutura de nuvem, o que deve ser considerado pelas organizações ao planejar a adoção de tais tecnologias de segurança.

Trabalhos futuros podem explorar a cobertura de outros cenários de ataques em aplicações hospedadas na AWS, e analisar as ferramentas disponibilizadas pelo provedor para cada caso distinto. Outras frentes de investigação incluem avaliar e comparar os serviços de segurança em nuvem de provedores como Google Cloud e Microsoft Azure, analisando a eficácia de suas soluções de mitigação em relação às oferecidas pela AWS.

Referências

AL-TALAK, K.; ABBASS, O. Detecting server-side request forgery (ssrf) attack by using deep learning techniques. *International Journal of Advanced Computer Science and Applications*, The Science and Information Organization, v. 12, n. 12, 2021. Disponível em: <<http://dx.doi.org/10.14569/IJACSA.2021.0121230>>. Citado na página 18.

ALMISHAL, A.; YOUSSEF, A. Cloud service providers: A comparative study. *International Journal of Computer Applications Information Technology*, v. 5, p. 2278–7720, 05 2014. Citado na página 9.

Amazon Web Services. *AWS Best Practices for DDoS Resiliency*. 2024. <<https://docs.aws.amazon.com/whitepapers/latest/aws-best-practices-ddos-resiliency/aws-best-practices-ddos-resiliency.html>>. Acesso em: 18 abr. 2025. Citado na página 16.

Amazon Web Services. *AWS WAF Developer Guide*. 2024. Disponível em: <<https://docs.aws.amazon.com/pdfs/waf/latest/developerguide/waf-dg.pdf>>. Acesso em: 18 abr. 2025. Citado na página 15.

Amazon Web Services. *Elastic Load Balancing: Application Load Balancers*. 2024. Disponível em: <<https://docs.aws.amazon.com/pdfs/elasticloadbalancing/latest/application/elb-ag.pdf>>. Acesso em: 18 abr. 2025. Citado 2 vezes nas páginas 14 e 15.

Amazon Web Services. *IAM Roles*. 2024. Disponível em: <https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html>. Acesso em: 18 abr. 2025. Citado na página 15.

ARMBRUST, M. et al. A view of cloud computing. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. ISSN 0001-0782. Disponível em: <<https://doi.org/10.1145/1721654.1721672>>. Citado na página 13.

BUYA, R. et al. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, v. 25, n. 6, p. 599–616, 2009. ISSN 0167-739X. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0167739X08001957>>. Citado na página 9.

CHECKPOINT. *Cloud Security Report*. 2024. Disponível em: <<https://engage.checkpoint.com/2024-cloud-security-report/items/2024-cloud-security-report-0e1c>>. Acesso em: 27 maio 2024. Citado 2 vezes nas páginas 10 e 11.

CHECKPOINT. *Cyber Security Report*. 2024. Disponível em: <<https://www.checkpoint.com/resources/items/2024-cyber-security-report>>. Acesso em: 27 maio 2024. Citado na página 10.

Cybersecurity and Infrastructure Security Agency. *Understanding Denial-of-Service Attacks*. 2024. Disponível em: <<https://www.cisa.gov/news-events/news/>>

[understanding-denial-service-attacks>](#). Acesso em: 18 abr. 2025. Citado 2 vezes nas páginas 13 e 16.

HALFOND, W. G. J.; VIEGAS, J.; ORSO, A. A classification of sql injection attacks and countermeasures. In: *International Symposium on Signals, Systems, and Electronics*. [s.n.], 2006. Disponível em: <https://api.semanticscholar.org/CorpusID:5969227>. Citado na página 17.

International Organization for Standardization; International Electrotechnical Commission. *ISO/IEC 27000:2018 — Information technology — Security techniques — Information security management systems — Overview and vocabulary*. 5. ed. Geneva: ISO/IEC, 2018. Disponível em: <https://www.iso.org/standard/73906.html>. Citado na página 12.

MELL, P. M.; GRANCE, T. *SP 800-145. The NIST Definition of Cloud Computing*. Gaithersburg, MD, USA, 2011. Citado 3 vezes nas páginas 9, 13 e 14.

OWASP Foundation. *Server-Side Request Forgery (SSRF)*. 2024. Disponível em: https://owasp.org/www-community/attacks/Server_Side_Request_Forgery. Acesso em: 18 abr. 2025. Citado na página 17.

OWASP Foundation. *SQL Injection*. 2024. Disponível em: https://owasp.org/www-community/attacks/SQL_Injection. Acesso em: 18 abr. 2025. Citado 2 vezes nas páginas 16 e 17.

PATIL, S. P. Enhancing cloud security by integrating data masking techniques with aws for effective ddos prevention. *International Journal of Intelligent Systems and Applications in Engineering*, v. 12, n. 22s, p. 748 –, Jul. 2024. Disponível em: <https://ijisae.org/index.php/IJISAE/article/view/6551>. Citado 4 vezes nas páginas 14, 15, 16 e 18.

POPOVIĆ, K.; HOCENSKI, Cloud computing security issues and challenges. In: *The 33rd International Convention MIPRO*. [S.l.]: MIPRO, 2010. p. 344–349. Citado na página 9.

RATH, A. et al. Security pattern for cloud saas: From system and data security to privacy case study in aws and azure. *Computers*, v. 8, n. 2, 2019. ISSN 2073-431X. Disponível em: <https://www.mdpi.com/2073-431X/8/2/34>. Citado na página 18.

SINGH, R. Cloud computing and data security : Aws and google case study. *International Journal of Computer Sciences and Engineering, IJCSE*, Indore, INDIA, v. 7, p. 164–171, 2 2019. ISSN 2347-2693. Disponível em: https://www.ijcseonline.org/full_paper_view.php?paper_id=3638. Citado na página 19.

STALLINGS, W.; BROWN, L. *Segurança de Computadores: Princípios e Práticas*. 2. ed. Rio de Janeiro: Elsevier, 2014. Tradução de: *Computer Security: Principles and Practice*, 2nd ed. ISBN 9788535264494. Citado 2 vezes nas páginas 12 e 13.

TASNIM, R. et al. *A Comparative Study on Three Selective Cloud Providers*. Academy and Industry Research Collaboration Center (AIRCC), 2022. 167–178 p. Disponível em: <http://dx.doi.org/10.5121/ijci.2022.110413>. Citado na página 9.

VENTURES, C. *Official Cybercrime Report*. 2023. Disponível em: <<https://cybersecurityventures.com/cybercrime-to-cost-the-world-9-trillion-annually-in-2024>>. Acesso em: 27 maio 2024. Citado na página 11.