Identificação Automática de *Buckets* de Armazenamento Vulneráveis com GENBUCKET

Milton Bazé[1], José Fabris[1], Fabrício S. de Paula[2], Carlos Alberto da Silva[1], Ronaldo A. Ferreira[1]

[1] Universidade Federal de Mato Grosso do Sul (UFMS)

[2]Universidade Estadual de Mato Grosso do Sul (UEMS)

milton.baze@ufms.br, jose.fabris@ufms.br, fabricio@comp.uems.br

carlos.silva@ufms.br, ronaldo.ferreira@ufms.br

Abstract. *This paper presents* GENBUCKET*, a modular tool for generating and validating cloud storage bucket names using modern generative models.* GENBUCKET *supports LSTM, Transformer, and GPT models trained on customizable datasets to capture diverse naming patterns. It automatically generates candidate names, verifies their existence via DNS, classifies them via HTTP, and analyzes public buckets for security vulnerabilities. Using validated data,* GENBUCKET *achieved hit rates of up to 21.73%, which is more than ten times the previous best-known result, and uncovered tens of buckets with vulnerabilities. By integrating generation, validation, and analysis,* GENBUCKET *enables automated detection of misconfigurations in cloud storage services.*

Resumo. *Este artigo apresenta* GENBUCKET*, uma ferramenta modular para geração e validação de nomes de buckets em nuvem com modelos generativos modernos.* GENBUCKET *suporta LSTM, Transformer e GPT, treinados com conjuntos de dados personalizáveis para capturar diferentes padrões de nomeação. A ferramenta gera automaticamente nomes candidatos, verifica suas existências via DNS, classifica-os via HTTP e analisa buckets públicos em busca de vulnerabilidades. Com dados validados,* GENBUCKET *alcançou até 21,73% de acerto, que é mais de dez vezes o melhor resultado conhecido, e identificou dezenas de buckets com vulnerabilidades. Ao integrar geração, validação e análise,* GENBUCKET *contribui para a detecção automatizada de falhas de configuração em serviços de nuvem.*

1. Introdução

A computação em nuvem oferece recursos computacionais sob demanda por meio de serviços de armazenamento, banco de dados, aplicações, ambientes de rede, máquinas virtuais, entre outros. Os provedores hospedam e controlam esses serviços, enquanto os clientes pagam apenas pelos recursos utilizados, eliminando a necessidade de investimentos em infraestrutura própria. Essa flexibilidade impulsionou o mercado de nuvem, cuja receita deve crescer 20,4% ao ano de 2025 a 2030, saltando de 752 bilhões para 2,39 trilhões de dólares [Research 2025].

O crescimento desse mercado, no entanto, aumenta a exposição de dados sensíveis. Segundo relatório de 2024 [Tenable 2024], 95% das organizações sofreram violações de segurança na nuvem nos 18 meses anteriores à pesquisa, e 92% relataram a exposição de dados confidenciais. Erros de configuração em serviços de armazenamento figuram entre as principais causas dessas violações.

Entre esses serviços, destaca-se o armazenamento de dados não estruturados em *buckets*, que são grandes recipientes acessados diretamente por meio de URLs (*Uniform Resource Locators*). Um *bucket* pode armazenar diversos objetos, como arquivos de vídeo, áudio, fotos, *logs* e *backups*. Apesar das boas práticas recomendadas pelos principais provedores - como AWS S3 [AWS 2025], Google Cloud Storage [Google 2025], Digital Ocean Spaces [Ocean 2025] e Microsoft Azure Storage [Microsoft 2023] - as políticas de acesso são definidas pelas organizações contratantes, muitas vezes sem pessoal especializado em segurança, o que leva a configurações incorretas e acessos indevidos. Além disso, os usuários de *buckets* muitas vezes armazenam dados sensíveis de aplicações, como nomes de usuário e senhas, ou confidenciais, como dados pessoais, expondo suas organizações a atores maliciosos [Eldad 2023, Westervelt 2013].

Um *bucket* pode ser configurado como *privado* ou *público*. No modo privado, o acesso exige autenticação e permissões específicas; no modo público, qualquer pessoa que conheça o URL pode acessar os dados. Incidentes causados por configurações incorretas têm comprometido a integridade, disponibilidade e privacidade de dados em larga escala, mesmo com a disponibilização de ferramentas de segurança pelos provedores. Alguns exemplos notáveis incluem:

- Em 2025, a Hipshipper expôs mais de 14 milhões de registros de clientes devido a uma configuração incorreta em um *bucket* S3 da Amazon [Mushtaq 2025];
- Em 2023, a Microsoft expôs acidentalmente 38TB de dados privados no GitHub devido à má configuração do mecanismo de compartilhamento SAS (*Shared Access Signatures*) da Azure [Ben-Sasson and Greenberg 2023];
- Em 2022, *buckets* da AWS S3 mal configurados vazaram 3TB de dados de aeroportos na Colômbia e no Peru [Cisoadvisor 2023];
- Entre 2021 e 2022, diversas falhas expuseram informações de 24,2 milhões de brasileiros [Surfshark 2022], incluindo dados de torcedores de futebol em *buckets* da empresa Futebol Card [Mari 2020].

Uma característica que facilita esses vazamentos é que os nomes dos *buckets* devem ser únicos no sistema global de armazenamento do provedor de nuvem, o que permite que agentes maliciosos usem técnicas de varredura ou força bruta para localizar *buckets* mal configurados. Diante desse cenário, é fundamental que provedores de serviços de Internet, empresas, universidades, equipes de auditoria e órgãos reguladores atuem de forma sistemática e proativa na identificação e correção dessas falhas visando promover boas práticas de segurança.

A ferramenta mais avançada atualmente para localização de *buckets* vulneráveis [Cable et al. 2021] emprega técnicas de análise de senhas combinadas com um modelo LSTM (*Long Short-Term Memory*) [Houdt et al. 2020]. Embora tenha sido pioneira no uso de modelos de aprendizado de máquina para essa tarefa, cada modelo

empregado obteve individualmente menos de 2% de acerto, além de a ferramenta não automatizar a verificação de vulnerabilidades.

Este trabalho apresenta a ferramenta GENBUCKET, que avança significativamente o estado da arte ao automatizar a geração de nomes de *buckets* e a verificação de suas vulnerabilidades utilizando modelos generativos modernos e flexíveis. Diferentemente das abordagens anteriores, GENBUCKET foi projetada para permitir a avaliação comparativa de diferentes modelos e estratégias de treinamento.

Especificamente, GENBUCKET integra nativamente três tipos de modelos:

- Um modelo LSTM tradicional.
- Um modelo, GPT-Neo-125M, baseado em GPT (*Generative Pre-trained Transformer*) [EleutherAI 2023], reconhecido pela sua capacidade de capturar padrões linguísticos complexos e gerar sequências altamente coerentes. O modelo foi pré-treinado com um grande corpus da Internet.
- Um modelo *Transformer* desenvolvido do zero, sem pré-treinamento, permitindo análises controladas de desempenho ao se variar parâmetros básicos do modelo.

Além de suportar diferentes arquiteturas generativas, GENBUCKET permite o uso de múltiplos conjuntos de treinamento. Inicialmente, a ferramenta é configurada com um corpus em português, composto por termos extraídos de dicionários, listas de senhas vazadas, gírias e nomes comuns no Brasil. Essa escolha visa capturar padrões linguísticos e morfológicos característicos do contexto brasileiro, ampliando a eficácia na descoberta de *buckets* nomeados em português. A ferramenta, no entanto, é totalmente parametrizável, possibilitando experimentos com diferentes idiomas e configurações de treinamento. Outro diferencial de GENBUCKET é a integração de um pipeline completo que não apenas gera nomes candidatos e verifica sua existência, mas também classifica os *buckets* encontrados como públicos ou privados e realiza análises automatizadas de segurança em *buckets* públicos.

Os resultados obtidos demonstram que modelos generativos, especialmente quando treinados com dados validados, são capazes de identificar milhares de *buckets* reais com taxas de acerto significativamente superiores às abordagens anteriores. O modelo GPT-Neo-125M, por exemplo, alcançou uma taxa de 19,05% de nomes únicos válidos, enquanto um modelo Transformer customizado superou essa marca com 21,73%, além de gerar nomes com menor sobreposição entre provedores. A análise de segurança revelou que uma fração expressiva dos *buckets* públicos apresentava vulnerabilidades reais, incluindo falhas críticas como execução remota de código. Esses achados reforçam o potencial de GENBUCKET como ferramenta eficaz tanto para geração quanto para auditoria automatizada de *buckets* em ambientes de armazenamento na nuvem.

Este trabalho está organizado da seguinte maneira. A Seção 2 apresenta a arquitetura modular de GENBUCKET e detalha o funcionamento dos seus módulos principais, como pré-processamento de dados, geração de nomes candidatos, verificação e classificação de *buckets*, análise de riscos e implementação. Na Seção 3, são apresentados os resultados da avaliação experimental de GENBUCKET. A Seção 4 compara GENBUCKET com os trabalhos existentes na literatura e a Seção 5 conclui este trabalho.

2. Arquitetura Modular da Ferramenta GENBUCKET

A ferramenta GENBUCKET é uma plataforma extensível para geração e validação de nomes de *buckets* em serviços de armazenamento em nuvem, bem como para análise de vulnerabilidades em objetos armazenados em *buckets* públicos. Seu principal objetivo é identificar automaticamente *buckets* públicos potencialmente mal configurados ou vulneráveis. No entanto, seu design modular permite substituir ou adaptar cada um de seus componentes principais. Altamente parametrizável, a ferramenta permite ao usuário explorar diferentes combinações de dados de treinamento (*e.g.,* com ou sem validação prévia), modelos generativos, variações nos parâmetros de geração e estratégias de análise de vulnerabilidades.

A Figura 1 apresenta o fluxo geral de execução de GENBUCKET, que pode ser dividido em quatro grandes módulos: *Pré-processamento de Dados*, *Geração de Nomes Candidatos*, *Verificação e Classificação de Buckets* e *Análise de Riscos e Vulnerabilidades*. Cada módulo expõe interfaces que permitem a experimentação com diferentes estratégias e implementações. A seguir, são descritas instâncias específicas utilizadas na implementação de referência.

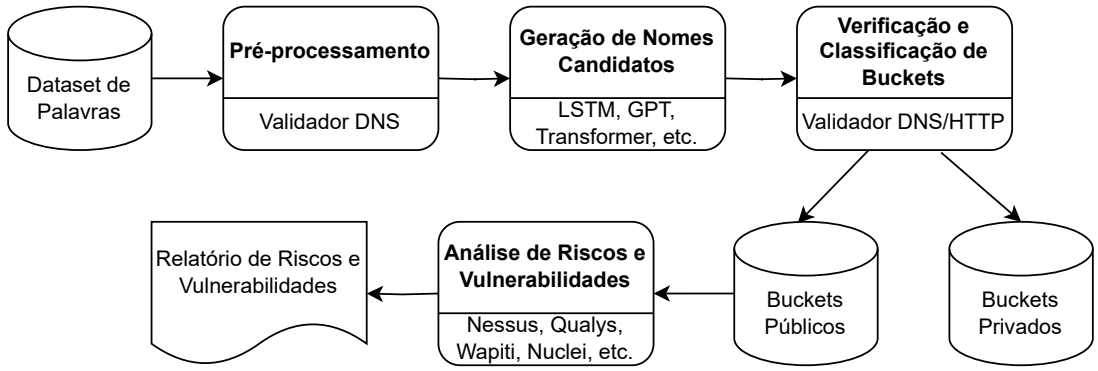

Figure 1. Arquitetura da ferramenta GENBUCKET.

2.1. Pré-processamento de Dados

O módulo de pré-processamento da ferramenta GENBUCKET prepara os dados utilizados no treinamento dos modelos generativos. O usuário pode optar por utilizar diretamente um conjunto de palavras fornecido ou aplicar uma etapa opcional de validação prévia, que verifica a existência de *buckets* com os nomes fornecidos por meio de consultas DNS. Essa validação explora o fato de que provedores de nuvem adotam padrões bem definidos para a nomeação completa de *buckets*. Por exemplo, o *bucket* `bucket_name` é acessado via `bucket_name.s3.amazonaws.com` na AWS S3 ou `bucket_name.storage.googleapis.com` no Google Cloud Storage.

Essa etapa, implementada pelo módulo Validador DNS da Figura 1, visa aumentar a qualidade do corpus de treinamento ao selecionar apenas nomes válidos efetivamente utilizados em ambientes reais. Com isso, busca-se capturar distribuições probabilísticas mais representativas dos padrões de nomenclatura empregados na criação de *buckets* ao invés de palavras provenientes de um dicionário genérico. Esse módulo permite que o usuário avalie o impacto de diferentes estratégias de pré-processamento na geração de nomes e o impacto das validações.

2.2. Geração de Nomes Candidatos

O módulo *Geração de Nomes Candidatos* de GENBUCKET é responsável por produzir sequências de caracteres (*i.e.,* palavras) com potencial para nomear *buckets* em provedores de nuvem. Essa geração é baseada em modelos generativos treinados previamente com conjuntos de dados preparados pelo módulo de pré-processamento.

Atualmente, GENBUCKET suporta três tipos de modelos: LSTM, Transformer e GPT. O modelo LSTM [Houdt et al. 2020] é adequado para capturar dependências sequenciais de curto e médio prazo e é eficaz em cenários com vocabulários compactos. Além disso, ele é o modelo generativo utilizado na melhor solução existente na literatura [Cable et al. 2021] e será adotado como referência de comparação para os demais. O modelo Transformer [Wolf et al. 2020], implementado do zero, explora mecanismos de atenção para capturar relações mais amplas no corpus de treinamento, permitindo melhor capacidade de generalização. Já o modelo GPT, incorporado à ferramenta GENBUCKET a partir de implementações pré-treinadas, também é baseado na arquitetura de transformer, mas é pré-treinado com um grande corpus de textos da Internet [EleutherAI 2023].

Tokenização por Caracteres. A tokenização adotada por GENBUCKET é baseada em caracteres, uma escolha essencial para tarefas de geração de nomes, em que a segmentação por palavras ou subpalavras é inadequada. Cada palavra do conjunto de treinamento é decomposta em uma sequência de caracteres individuais, o que permite capturar sua estrutura morfológica interna. Essa abordagem é especialmente eficaz em domínios compostos por identificadores curtos e frequentemente ausentes de dicionários, além de evitar a explosão do vocabulário e lidar naturalmente com palavras inéditas. Cada caractere é mapeado para um índice em um vocabulário de tamanho n, correspondente ao número de caracteres distintos no conjunto de treinamento. A sequência resultante é convertida em uma série de vetores representando prefixos crescentes da palavra: o primeiro vetor contém apenas o caractere inicial, e os seguintes são construídos com a adição progressiva de novos caracteres. A Figura 2 ilustra esse processo para a palavra "treino", considerando um alfabeto reduzido com os caracteres da própria palavra. Esses vetores alimentam os modelos generativos e, em seguida, são convertidos em representações densas por meio de *embeddings*.

```
Palavra=treino Alfabeto={``, e=1, i=2, n=3, o=4, r=5, t=6}

        Amostras de Treinamento          Rótulo
t:           [6, 0, 0, 0, 0, 0, 0, 0] -> 5, {r}
t r:         [6, 5, 0, 0, 0, 0, 0, 0] -> 1, {e}
t r e:       [6, 5, 1, 0, 0, 0, 0, 0] -> 2, {i}
t r e i:     [6, 5, 1, 2, 0, 0, 0, 0] -> 3, {n}
t r e i n:   [6, 5, 1, 2, 3, 0, 0, 0] -> 4, {o}
t r e i n o: [6, 5, 1, 2, 3, 4, 0, 0] -> 0, {``}
```

Figure 2. Tokenização da palavra "treino" e representação das amostras em vetores de dimensão oito (tamanho máximo de palavra).

Durante a inferência, *i.e.,* geração, o modelo utiliza a sequência de caracteres gerados até o momento como contexto para prever o próximo caractere, construindo o nome de forma autoregressiva – isto é, com base exclusivamente na sequência prévia.

Arquitetura do Modelo LSTM. A arquitetura LSTM utilizada por GENBUCKET é composta por três camadas principais: entrada, camada LSTM e saída. A entrada recebe sequências de caracteres representadas por vetores esparsos codificados com o processo exemplificado na Figura 2.

A camada central é uma LSTM unidirecional com h unidades ocultas, responsável por processar a sequência de vetores de entrada e capturar dependências contextuais entre os caracteres. Essa camada mantém um estado interno que é atualizado a cada caractere lido, permitindo que o modelo aprenda padrões como sufixos, prefixos, ou combinações frequentes. Durante o treinamento, é comum utilizar sequências de comprimento fixo T, extraídas com janelas deslizantes sobre o conjunto de nomes tokenizados. Em outras palavras, o rótulo de um vetor de entrada, correspondente a uma sequência de caracteres, é simplesmente o caractere subsequente ao último caractere da sequência representada pelo vetor.

A camada de saída é uma camada densa totalmente conectada com ativação *Softmax*, que projeta o estado oculto da LSTM em uma distribuição de probabilidade sobre os n caracteres do vocabulário. O caractere de maior probabilidade é então selecionado (ou amostrado probabilisticamente, dependendo da temperatura configurada) e concatenado à sequência atual durante a inferência.

Entre os principais parâmetros configuráveis do modelo LSTM estão: o número de unidades ocultas (h), o comprimento das sequências de entrada (T), a função de perda (*e.g., sparse categorical crossentropy*), o otimizador (*e.g.,* Adam), a temperatura de amostragem e o número de nomes candidatos a serem gerados. O treinamento utiliza preenchimento (*padding*) para uniformizar o comprimento das sequências. Esses parâmetros influenciam diretamente a diversidade e a qualidade dos nomes gerados, permitindo que GENBUCKET produza candidatos realistas e representativos dos padrões observados em *buckets* reais.

Arquitetura do Modelo Transformer. A ferramenta GENBUCKET também oferece suporte a um modelo de linguagem baseado na arquitetura Transformer, implementado do zero com auxílio da biblioteca PyTorch [Paszke et al. 2019]. Esse modelo segue uma estrutura inspirada no GPT, composta por camadas de *embedding*, atenção multi-cabeça (*Multi-Head Attention*), camadas *feed-forward* posicionais, normalização (*LayerNorm*) e uma camada de saída com ativação *Softmax* [Vaswani et al. 2017].

A entrada consiste em sequências de caracteres tokenizadas individualmente. Cada caractere é mapeado para um índice numérico e representado por um vetor denso por meio de *embeddings* aprendidos, somados a vetores posicionais para preservar a ordem dos símbolos. O modelo emprega atenção autoregressiva mascarada, garantindo que a predição de um caractere leve em conta apenas os anteriores na sequência.

Durante a geração, o modelo parte de uma sequência de caracteres inicial, que pode conter um único caractere, e prevê iterativamente os próximos símbolos, formando nomes completos de forma autoregressiva. Apenas nomes únicos são mantidos, sendo normalizados (por exemplo, remoção de símbolos e conversão para minúsculas) antes de serem armazenados.

Entre os principais hiperparâmetros estão: o tamanho dos *embeddings*, o número de cabeças de atenção, a profundidade da arquitetura, e a temperatura de amostragem. O

modelo é treinado com o otimizador Adam e função de perda por entropia cruzada (*cross-entropy*). A arquitetura permite capturar dependências de longo alcance entre caracteres, gerando nomes diversos e coerentes com os padrões observados no dataset de treinamento (*e.g.,* nomes representando *buckets* reais).

Arquitetura do Modelo GPT. A ferramenta GENBUCKET também inclui um modelo GPT, mais especificamente o modelo GPT-Neo-125M, uma arquitetura de linguagem baseada em Transformer, pré-treinada com um grande corpus de textos da Internet. Desenvolvido pela EleutherAI [EleutherAI 2023], esse modelo segue a mesma arquitetura unidirecional do GPT-2, com camadas empilhadas de atenção autoregressiva e camadas *feed-forward*, permitindo geração de texto fluente e coerente a partir de um contexto inicial.

Diferentemente dos modelos anteriores, o GPT-Neo-125M é inicialmente treinado em larga escala com dados genéricos de linguagem natural. Para adaptá-lo à tarefa de geração de nomes de *buckets*, GENBUCKET aplica um processo de ajuste fino (*fine-tuning*) com os dados específicos do domínio: um conjunto de palavras tokenizadas por caractere fornecidas como entrada pelo usuário. Esse ajuste permite que o modelo aprenda a gerar sequências compatíveis com os padrões estruturais e estatísticos presentes nesse tipo de nomenclatura para reduzir a tendência a produzir texto genérico ou fora do domínio.

Durante a geração, o modelo é alimentado com um contexto inicial representado por uma sequência de caracteres e produz novas palavras de forma autoregressiva, um símbolo por vez, até atingir o critério de parada. As palavras geradas passam por filtros de unicidade e normalização, como nos modelos anteriores.

Os principais parâmetros configuráveis para o GPT-Neo-125M incluem a temperatura de amostragem, o comprimento máximo das sequências geradas e o número de amostras. Como se trata de um modelo com capacidade significativamente maior que os anteriores, ele tende a produzir nomes mais variados, embora exija maior custo computacional durante o processo de geração.

2.3. Verificação e Classificação de Buckets

O módulo de *Verificação e Classificação de Buckets* de GENBUCKET tem como objetivo verificar a existência de um *bucket* e distinguir entre *buckets* públicos e privados. A classificação é realizada por meio de requisições HTTP anônimas, sem uso de credenciais, simulando o comportamento de um atacante ou agente externo.

Para cada *bucket* válido, a ferramenta emite requisições `GET` ou `HEAD` para URLs padronizados, como `https://<bucket>.s3.amazonaws.com` na AWS S3 ou `https://storage.googleapis.com/<bucket>` no Google Cloud Storage. As respostas HTTP são analisadas para inferir o nível de exposição do *bucket*. Por exemplo:

- Uma resposta HTTP 200 com conteúdo listável ou arquivos acessíveis indica que o *bucket* é público;
- Uma resposta HTTP 403 (“Access Denied”) geralmente indica que o *bucket* existe, mas não permite acesso anônimo aos objetos, caracterizando um *bucket* privado. Em alguns casos, a mensagem de erro 403 pode conter detalhes adicionais, como `<Code>AllAccessDisabled</Code>` ou

`<Code>AccessDenied</Code>`, que são úteis para distinguir políticas mais ou menos restritivas;
- Uma resposta HTTP 404 indica que o *bucket* não existe ou está corretamente configurado para não revelar sua existência a usuários não autenticados;
- Uma resposta HTTP 400 indica que o nome do *bucket* é inválido.

O módulo também registra metadados relevantes retornados nas respostas, como cabeçalhos HTTP e códigos de erro específicos, permitindo análises mais detalhadas sobre políticas de acesso e configurações de segurança dos *buckets*.

2.4. Análise de Riscos e Vulnerabilidades

O módulo de análise de riscos e vulnerabilidades da ferramenta GENBUCKET tem como objetivo identificar *buckets* públicos potencialmente inseguros para avaliar seus níveis de exposição e presença de falhas que representem riscos concretos à segurança da informação. Uma vez classificados como públicos, os *buckets* são submetidos a testes automatizados por meio de três estratégias complementares: avaliação de permissões, análise de objetos web e análise de vulnerabilidades com ferramentas de apoio.

Avaliação de Permissões e Exposição de Dados. A primeira estratégia considera como vulneráveis os *buckets* que permitem acesso público a informações sensíveis ou que apresentam permissões inadequadas de escrita ou exclusão de objetos. Os testes conduzidos nessa etapa incluem:

- **Listagem de objetos**: verificação da possibilidade de listar o conteúdo do *bucket* por meio de requisições HTTP anônimas;
- **Leitura de arquivos**: tentativas de acesso direto a objetos armazenados, sem necessidade de autenticação;
- **Escrita controlada**: tentativa segura de upload de objetos com o objetivo de inferir permissões de escrita, sem persistência ou impacto no ambiente.

Esses testes seguem boas práticas éticas e legais, assegurando que nenhuma ação comprometa a integridade dos dados avaliados. Quando qualquer uma dessas operações é permitida sem autenticação, o *bucket* é marcado como potencialmente vulnerável. Foram realizados apenas testes controlados em *buckets* públicos, acessíveis sem autenticação e sem violar os termos de uso das plataformas. Não foram realizados *downloads* de conteúdos, mas apenas verificações de metadados. Nos casos em que foi possível gravar arquivos, foram deixadas mensagens de alerta nos *buckets* vulneráveis para seus responsáveis.

Análise de Objetos Web e Superfícies de Ataque. A segunda estratégia foca em *buckets* que armazenam arquivos associados a aplicações web, especialmente aqueles com extensões como `.js`, `.ts`, `.html`, `.php`, `.json`, `.svg`, entre outras. O objetivo é identificar:

- **Código exposto**: arquivos contendo lógica de negócio ou segredos embutidos, como chaves de API;
- **Erros de configuração**: arquivos que revelam detalhes internos do sistema, caminhos de diretórios ou comentários sensíveis;
- **Superfícies vulneráveis**: componentes com falhas que possam ser exploradas por atacantes.

Ferramentas de Apoio à Análise. GENBUCKET integra um conjunto de ferramentas automatizadas para analisar vulnerabilidades em *buckets* públicos. Inicialmente, uma ferramenta desenvolvida em Python [Weaver 2017] permite interagir com *buckets* AWS S3 para testar operações como listagem, leitura e escrita de objetos. Além disso, são utilizados os seguintes *scanners* públicos:

- **Nessus Professional** [Tenable, Inc. 2024]: identifica falhas conhecidas em configurações, serviços expostos, bibliotecas vulneráveis e permissões excessivas.
- **Qualys WAS** [Qualys, Inc. 2025]: simula ataques a aplicações web para detectar injeções de código (como SQLi e XSS), falhas de autenticação e outros vetores de ataque.
- **Wapiti** [Surribas 2006]: realiza varreduras em aplicações web acessíveis a partir de *buckets*, detectando vulnerabilidades com base em pontos de entrada como formulários e parâmetros HTTP.
- **Nuclei** [ProjectDiscovery 2024]: executa testes de reconhecimento e exploração de falhas conhecidas usando templates configuráveis, sendo eficaz para análise em larga escala com alto grau de personalização.

Com o módulo de análise de riscos e vulnerabilidades, GENBUCKET vai além da geração de nomes de *buckets*, atuando como uma ferramenta automatizada para descoberta e validação de falhas de segurança em ambientes reais de armazenamento em nuvem. Devido à sua arquitetura modular, GENBUCKET pode ser facilmente estendida para incorporar novos testes, heurísticas e ferramentas de varredura, permitindo sua adaptação a diferentes provedores de nuvem, tipos de vulnerabilidades ou requisitos específicos de análise de segurança.

2.5. Implementação e Extensão

GENBUCKET é implementada em Python 3 e define classes virtuais para os módulos listados na Figura 1 que podem ser redefinidos para incorporar novos métodos de pré-processamento, modelos generativos e verificação e classificação de *buckets*. Para incorporar um novo modelo generativo, por exemplo, é necessário simplesmente implementar uma nova classe com os métodos `fit` and `predict`. No caso de análise de vulnerabilidades, GENBUCKET invoca programas externos e pode ser estendida definindo-se novos programas no arquivo de configuração da ferramenta. O código-fonte de GENBUCKET bem como os datasets utilizados para avaliação experimental discutida na Seção 3 estão disponíveis em [Bazé et al. 2025].

3. Avaliação Experimental

Esta seção apresenta uma avaliação experimental do uso de modelos generativos para a geração de nomes representativos de *buckets* de armazenamento em nuvem, bem como para identificação de riscos e vulnerabilidades associadas a *buckets* públicos detectados a partir dos nomes gerados. GENBUCKET é utilizada como plataforma para treinar, gerar e validar nomes com base em diferentes conjuntos de dados e modelos, incluindo LSTM, GPT-Neo-125M e Transformer. A avaliação tem dois focos: *(i)* analisar o impacto da qualidade dos dados de treinamento na geração de nomes representativos de *buckets* e *(ii)* explorar o uso de GENBUCKET como ferramenta de auditoria automatizada de segurança.

Table 1. Médias e margens de erro de *buckets* encontrados com treinamento dos modelos usando dados *não* validados.

	Buckets	AWS S3	Google Storage	Digital Ocean	Média Geral	Média Únicos
LSTM	Todos	366,0 ± 15,3	217,6 ± 10,7	12,7 ± 2,6	596,3 ± 21,2	511,5 ± 18,8
	Privados	354,2 ± 14,9	183,7 ± 9,4	11,4 ± 2,3	549,3 ± 20,7	465,2 ± 18,2
	Públicos	11,8 ± 3,2	33,9 ± 5,1	1,3 ± 1,2	47,0 ± 4,8	46,3 ± 4,6
GPT	Todos	386,7 ± 76,6	379,2 ± 77,9	43,5 ± 8,9	809,4 ± 163,0	573,7 ± 101,2
	Privados	363,2 ± 76,6	362,8 ± 77,9	38,6 ± 8,9	764,6 ± 162,9	532,6 ± 101,0
	Públicos	23,5 ± 2,8	16,4 ± 2,2	4,9 ± 0,4	44,8 ± 4,5	41,1 ± 4,8

3.1. Conjuntos de Dados

Para avaliar o desempenho dos modelos generativos na criação de nomes representativos de *buckets*, foram utilizados dois conjuntos de dados distintos. O primeiro, com 312.010 palavras, foi compilado a partir de fontes públicas como listas do Broffice [BR-Office 2016], dicionários de senhas vazadas [NordSecurity 2023], além de gírias, nomes próprios e sobrenomes comuns no Brasil [Donda 2018]. O uso de um dicionário em português permite simular cenários mais realistas no contexto brasileiro, refletindo padrões de nomeação adotados por usuários locais e testando a capacidade dos modelos em gerar nomes plausíveis em um idioma não explorado por trabalhos anteriores. Esse conjunto foi utilizado diretamente no treinamento dos modelos. Alternativamente, a mesma base foi submetida a uma validação por meio de consultas DNS em três provedores (AWS, Google Cloud e DigitalOcean), resultando em um subconjunto com 24.325 nomes efetivamente existentes. Ambos os conjuntos foram empregados nos experimentos com o objetivo de analisar o impacto da qualidade dos dados na geração de nomes e na descoberta de *buckets* reais.

3.2. Configuração dos Experimentos

Foram conduzidos dois experimentos: um com o conjunto completo de palavras (não validado) e outro com o conjunto validado. Os modelos LSTM e GPT-Neo-125M foram treinados com ambos os conjuntos, enquanto o modelo Transformer foi avaliado apenas com o conjunto validado, dado seu melhor desempenho nos dois primeiros modelos. Essa separação permite analisar o impacto da qualidade e relevância dos dados de treinamento na geração de nomes eficazes.

Cada modelo gerou automaticamente 10 conjuntos de 10.000 nomes distintos entre si e distintos das palavras utilizadas no treinamento, totalizando 100.000 nomes por modelo e assegurando que os nomes gerados não fossem mera repetição dos dados originais. Esses nomes foram então verificados por meio de requisições DNS aos serviços da AWS S3, Google Cloud Storage e DigitalOcean com o objetivo de identificar quais nomes correspondiam a *buckets* existentes. Os *buckets* encontrados foram classificados como públicos ou privados com base nas respostas dos servidores a requisições HTTP e nas permissões observadas.

3.3. Resultados da Geração e Validação de Buckets

A Tabela 1 apresenta os resultados dos modelos LSTM e GPT com o conjunto de dados não validados. As linhas da tabela mostram as médias e as respectivas margens de erro,

Table 2. Médias e margens de erro de *buckets* encontrados com treinamento dos modelos usando dados validados.

	Buckets	AWS S3	Google Storage	Digital Ocean	Média Geral	Média Únicos
LSTM	Todos	1831,7 ± 215,6	1265,1 ± 189,8	125,4 ± 23,5	3222,2 ± 426,1	1572,6 ± 158,4
	Privados	1752,0 ± 214,6	1134,5 ± 188,6	109,8 ± 23,3	2996,3 ± 424,5	1433,0 ± 157,3
	Públicos	79,7 ± 13,1	130,6 ± 21,4	15,6 ± 3,1	225,9 ± 36,8	139,6 ± 18,7
GPT	Todos	2163,4 ± 371,5	1851,0 ± 378,4	290,4 ± 88,5	4304,8 ± 837,0	1904,6 ± 301,1
	Privados	2040,7 ± 370,2	1718,6 ± 377,6	253,5 ± 87,2	4012,8 ± 834,2	1715,6 ± 297,9
	Públicos	122,7 ± 30,3	132,4 ± 24,6	36,9 ± 15,5	292,0 ± 68,2	188,1 ± 44,3

para intervalos de confiança com nível de confiança de 95%, das quantidades totais de *buckets* identificados, bem como a distribuição entre *buckets* privados e públicos.

Com esse conjunto, o modelo LSTM identificou em média 596,3 *buckets*, sendo 511,5 nomes únicos. Em termos de classificação, foram 549,3 *buckets* privados e 47,0 públicos, em média. A taxa de acerto considerando apenas nomes únicos foi de 5,12% – mais que o dobro da melhor ferramenta anterior [Cable et al. 2021], que alcança menos de 2% mesmo com dados validados. Como nesses dois casos são utilizados modelos do tipo LSTM, o principal diferencial está na qualidade do conjunto de treinamento utilizado neste trabalho, que foi mais eficaz em capturar as distribuições de sequências de caracteres usadas em nomes reais de *buckets*. O modelo GPT, por outro lado, obteve desempenho ainda melhor, com um aumento de 12,16% no número médio de nomes únicos (573,7 contra 511,5) e de 35,7% no total de *buckets* identificados (809,4 contra 596,3), considerando possíveis repetições de nomes entre provedores.

A Tabela 2 apresenta os resultados obtidos com o conjunto de dados validados, revelando um impacto expressivo na eficácia da geração de nomes. As médias de nomes únicos encontrados mais que triplicaram em relação à base não validada: passaram de 511,5 para 1.572,6 com o modelo LSTM, e de 573,7 para 1.904,6 com o modelo GPT. Isso se reflete diretamente nas taxas médias de acerto, que atingiram 15,72% para o LSTM e 19,05% para o GPT. Este último resultado representa um aumento de mais de nove vezes em relação ao melhor trabalho conhecido [Cable et al. 2021]

Esses resultados demonstram que os nomes realmente utilizados em *buckets* seguem padrões estruturais específicos que podem ser aprendidos por modelos generativos, especialmente os mais sofisticados. Isso reforça a importância de se adotar políticas de acesso mais restritivas e mecanismos de monitoramento contínuo, já que atacantes também podem empregar modelos semelhantes para identificar e explorar *buckets* expostos em larga escala.

3.3.1. Avaliação do Modelo Transformer

Um modelo baseado na arquitetura Transformer oferece diversos hiperparâmetros que podem ser ajustados para melhorar seu desempenho. No entanto, modelos pré-treinados como o GPT-Neo-125M, utilizado na Seção 3.3, geralmente têm parâmetros internos fixos e permitem apenas ajustes limitados, como o re-treinamento com novos dados. Para investigar de forma mais detalhada o impacto de parâmetros internos, esta seção apresenta os resultados obtidos com um modelo Transformer implementado do zero.

Table 3. Resultados obtidos para *buckets* com Transformer por embeddings .

Embedding	Tipo	Média Geral	Média Únicos
16	Todos	2147,9 ± 393,5	1564,2 ± 242,6
	Privados	2028,9 ± 393,4	1447,3 ± 241,8
	Públicos	119,0 ± 8,8	116,9 ± 20,12
32	Todos	2468,9 ± 399,4	1818,4 ± 243,1
	Privados	2332,7 ± 398,5	1685,2 ± 241,8
	Públicos	136,2 ± 26,5	133,2 ± 25,9
40	Todos	2380,8 ± 332,8	1749,5 ± 203,6
	Privados	2246,3 ± 332,6	1618,8 ± 202,8
	Públicos	134,5 ± 18,3	130,9 ± 17,4
64	Todos	2583,1 ± 384,3	1909,5 ± 238,4
	Privados	2428,3 ± 383,5	1758,0 ± 237,3
	Públicos	154,8 ± 24,3	151,5 ± 24,1
128	Todos	2951,0 ± 403,8	2173,6 ± 250,6
	Privados	2786,2 ± 403,3	2012,3 ± 249,8
	Públicos	164,8 ± 20,5	161 ± 19,6

O principal parâmetro avaliado foi a dimensão do vetor de *embedding*, cuja variação influencia diretamente a capacidade do modelo de representar relações entre os caracteres. A Tabela 3 mostra que o aumento dessa dimensão resulta em melhores taxas de acerto na geração de nomes, indicando que *embeddings* mais ricos permitem ao modelo capturar padrões estruturais com maior precisão.

O resultado mais significativo foi obtido com *embedding* de dimensão 128, que alcançou uma média de 2.173,6 nomes únicos válidos – um aumento de 14,12% em relação ao modelo GPT-Neo-125M (1.904,6). Esse desempenho não apenas supera o GPT-Neo-125M, como também evidencia o potencial de Transformers customizados quando configurados adequadamente. Outro aspecto relevante é que os nomes gerados pelo modelo construído do zero apresentaram menor sobreposição entre provedores de nuvem, o que sugere maior diversidade na geração e levanta novas hipóteses sobre a influência do treinamento e da arquitetura na generalização entre provedores – um tema promissor para investigações futuras, que também pode incluir a investigação de outros parâmetros da arquitetura.

3.4. Análise de Riscos e Vulnerabilidades

A avaliação de segurança dos *buckets* públicos identificados foi conduzida por duas estratégias complementares. A primeira consistiu na tentativa ética e controlada de gravação de objetos em *buckets* públicos, utilizando uma ferramenta baseada em Python [Weaver 2017]. Consideramos vulneráveis os *buckets* que permitiram escrita sem autenticação.

Os resultados mostraram que, entre os 2.259 *buckets* públicos gerados pelo LSTM, 62 (2,74%) permitiram escrita: 51 na AWS, 7 no Google Storage e 4 na DigitalOcean. Para o GPT, 72 de 2.920 *buckets* (2,47%) foram classificados como vulneráveis. O modelo Transformer (*embedding* 128) encontrou 55 *buckets* vulneráveis: 43 na AWS, 9 no Google e 3 na DigitalOcean.

A segunda estratégia focou na inspeção de *buckets* com arquivos voltados para aplicações web. Foram filtrados arquivos com extensões como `.js`, `.php`, `.html`,

`.json`, entre outras. No total, foram identificados 532 *buckets* contendo tais arquivos: 195 gerados pelo LSTM e 337 pelo GPT. As ferramentas Nessus [Tenable, Inc. 2024], Qualys WAS [Qualys, Inc. 2025], Wapiti [Surribas 2006] e Nuclei [ProjectDiscovery 2024] foram empregadas com configurações restritivas e controle de intrusão, a fim de preservar a integridade dos serviços. O Nessus foi configurado com *rate limit*, desativação de plugins destrutivos e mecanismos de contenção. Nuclei e Wapiti foram utilizadas para identificação de arquivos expostos e possíveis falhas de segurança em aplicações web.

Foram identificados 57 *buckets* com vulnerabilidades reais: 15 em *buckets* gerados pelo modelo LSTM e 42 pelo GPT. As falhas incluem CVEs como BEAST (CVE-2011-3389), SWEET32 (CVE-2016-2183) e uma *WebShell* JSP com execução remota de código (RCE) confirmada pelo VirusTotal [VirusTotal 2022]. As informações abaixo entre colchetes indicam o nível de risco e o número de *buckets* afetados – por exemplo, [4,3 MEDIUM; 10] indica risco médio e 10 *buckets* com essa falha. O total de ocorrências é superior ao número de *buckets* vulnerábeis, pois um mesmo *bucket* pode apresentar múltiplas vulnerabilidades.

- **CVE-2011-3389 (BEAST) [4,3 MEDIUM; 40]:** vulnerabilidade crítica no SSL/TLS 1.0 que permite interceptação de dados criptografados, comprometendo a confidencialidade da comunicação.
- **CVE-2001-1371 [7,5 HIGH; 7]:** falha de configuração no *Oracle Application Server*, permitindo que usuários anônimos implantem aplicativos via SOAP, o que facilita a execução remota não autorizada.
- **CWE-327 [7,5 HIGH; 40]:** uso de algoritmos criptográficos fracos, como RC4 ou 3DES, representando risco de quebra de criptografia por atacantes.
- **CVE-2016-2183 (SWEET32) [7,5 HIGH; 40]:** relacionada à CWE-327, trata-se da utilização de blocos criptográficos vulneráveis a ataques por colisão.
- **CVE-1999-0678 / CWE-276 [5,0 MEDIUM; 1]:** permite que usuários anônimos façam login via FTP com permissões de escrita, facilitando o upload de arquivos maliciosos.
- **CVE-2001-1446 [7,5 HIGH; 4]:** falha que permite ataques de negação de serviço (DoS) por meio do envio de pacotes malformados.
- **CVE-2002-2370 [5,0 MEDIUM; 1]:** vulnerabilidade no servidor *Simple Web Server (SWS)* que causa falhas no serviço via requisições malformadas.
- **WebShell JSP (RCE) [9,8 CRITICAL; 4]:** O arquivo, encontrado em um dos *buckets* públicos, foi examinado localmente e posteriormente enviado ao serviço VirusTotal [VirusTotal 2022], que confirmou se tratar de um malware com capacidade de execução remota de código (Remote Code Execution - RCE).

3.5. Conclusão da Avaliação

A avaliação experimental demonstrou que o uso de bases validadas para o treinamento de modelos generativos aumenta significativamente a taxa de acerto na geração de nomes de *buckets* válidos. Além disso, os experimentos evidenciam a superioridade do modelo GPT sobre o LSTM, e o impacto positivo do uso de *embeddings* maiores no Transformer. Por fim, as duas estratégias de análise de segurança permitiram a identificação de falhas concretas em *buckets* públicos, o que reforça o potencial de GENBUCKET como ferramenta automatizada de geração, validação e auditoria de segurança em ambientes de

armazenamento em nuvem. Devido à sua arquitetura modular, GENBUCKET pode ser facilmente estendida com novos testes, heurísticas e ferramentas de análise.

4. Trabalhos Relacionados

Trabalhos que lidam com os riscos de segurança associados a *buckets* em serviços de nuvem concentram-se em duas frentes: identificação de novos *buckets* e análise de vulnerabilidades.

Identificação de Novos Buckets. A identificação de *buckets* geralmente envolve a geração de nomes candidatos e verificação de suas existências, o que pode ser feito via formação de URLs ou consultas DNS. No entanto, dado que os provedores aceitam nomes de 3 a 63 caracteres, o espaço de busca pode atingir cerca de 10^{101} combinações [Cable et al. 2021], tornando a tarefa altamente desafiadora.

[Wood 2011] é o primeiro trabalho a realizar uma análise prática da exposição de dados em *buckets* do Amazon S3, partindo da hipótese de que usuários frequentemente utilizam seus próprios nomes ou sobrenomes. A partir de uma lista com 2.268 nomes, o trabalho identifica 131 *buckets* públicos e 848 privados, servindo como um alerta inicial sobre os riscos de configuração inadequada. De forma semelhante, [Willis 2013] busca identificar *buckets* com nomes de empresas, utilizando fontes como a lista Fortune 1000, domínios do Alexa Top Sites e APIs da Microsoft Azure. Os nomes candidatos são gerados por concatenação de sufixos e prefixos (*e.g.,* , -backup", www", .com", .net"). A abordagem resultou em 126 bilhões de combinações e levou à descoberta de 12.328 *buckets*, sendo 1.951 públicos. Uma amostra com 40 mil arquivos expostos revelou dados sensíveis, como fotos pessoais, registros de vendas e código-fonte. [Continella et al. 2018] automatizam o processo de geração e verificação de nomes, não se limitando a nomes predefinidos. Os nomes candidatos são formados por palavras curtas, acrônimos, mutações de palavras do inglês, e domínios. Além disso, o trabalho aplica rastreamento web para extrair URLs e usa DNS passivo, mapeando endereços IP da Amazon. No total, o trabalho identificou 240.461 *buckets* distintos (34.145 públicos). A ferramenta também realiza testes de segurança que identificam vulnerabilidades como vazamento de dados, injeção de código e desfiguração de sites.

Stratosphere [Cable et al. 2021] introduz uma abordagem baseada em técnicas de quebra de senhas, explorando a similaridade entre os padrões de nomeação de *buckets* e senhas. O trabalho utiliza três modelos: Probabilistic Context-Free Grammar (PCFG) [Weir et al. 2009], N-grams [Kelley et al. 2012] e LSTM [Houdt et al. 2020]. Os dados de treinamento foram obtidos a partir de repositórios públicos, como GitHub, e de fontes de DNS passivo. Os resultados mostram que Stratosphere encontrou 2,7 vezes mais *buckets* públicos, 5,8 vezes mais *buckets* mal configurados e 5,3 vezes mais *buckets* contendo documentos sensíveis em relação a [Continella et al. 2018]. GENBUCKET, por outro lado, utiliza modelos generativos mais modernos para a geração de nomes de *buckets* e obtem taxas mais elevadas de acerto que os trabalhos existentes.

Análise de Vulnerabilidades. [Yadmani et al. 2025] analisam empiricamente a exposição de segredos em *buckets* públicos acessíveis via a plataforma GrayhatWarfare [Grayhatwarfare 2018]. O trabalho buscou arquivos com extensões comuns no armazenamento de credenciais, como `.bat`, `.env`, `.json` e `.yml` e identificou 215 arquivos contendo segredos, e após divulgação responsável, 95 foram

corrigidos pelos proprietários ou provedores. [Bouchet et al. 2020] apresentam *Block Public Access*, uma ferramenta integrada à AWS para verificar automaticamente se as políticas de acesso de *buckets* permitem acesso público não confiável. O sistema bloqueia tais configurações e emite alertas ao usuário. [Baras et al. 2019] comparam os mecanismos de controle de acesso e segurança entre Amazon S3 e Azure Blob Storage, avaliando criptografia, IAM e integração com diretórios de identidade. O trabalho conclui que ambos os serviços mostraram infraestrutura robusta, sendo a escolha influenciada por requisitos específicos de conformidade ou integração.

Apesar dos avanços, os trabalhos mencionados ainda enfrentam limitações quanto à taxa de acerto e cobertura dos modelos. Isso motiva novas abordagens baseadas em aprendizado profundo e dados validados, como as exploradas neste artigo, além da necessidade de integração da geração de nomes com ferramentas de análise de vulnerabilidades, como a apresentada por GENBUCKET.

5. Conclusão

Este trabalho apresentou a ferramenta GENBUCKET e uma avaliação experimental detalhada do uso de modelos generativos para geração e validação de nomes de *buckets* em nuvem. Os resultados mostram que o uso de dados validados aumenta significativamente a taxa de acerto na descoberta de *buckets* reais, com destaque para o modelo GPT-Neo-125M (19,05%) e o Transformer com *embedding* 128 (21,73%). Além disso, a ferramenta identificou vulnerabilidades críticas em centenas de *buckets* públicos, incluindo permissões indevidas e arquivos perigosos. Esses achados reforçam o potencial de GENBUCKET como solução eficaz e automatizada para auditoria de segurança em serviços de armazenamento na nuvem.

Agradecimentos

O presente trabalho foi realizado com apoio da CAPES – Código de Financiamento 001, do CNPq – Procs. 420934/2023-5, 308101/2022-7 e 465446/2014-0, e da FAPESP – Procs. 2023/00812-7, 2023/00811-0, 2020/05192-9 e 2020/05183-0.

Uso de Inteligência Artificial (IA) Generativa

Este trabalho utilizou a ferramenta de IA Generativa ChatGPT somente para revisar alguns parágrafos do texto do artigo.

References

AWS (2025). Amazon S3 - Armazenamento de Objetos Construído para Armazenar e Recuperar Qualquer Volume de Dados. `http://www.aws.amazon.com/`.

Baras, S., Saeed, I., and Hajjdiab, H. (2019). Security and Privacy of AWS S3 and Azure Blob Storage Services. In *Proc. of 2019 IEEE ICCCS*.

Bazé, M., Fabris, J., de Paula, F. S., da Silva, C. A., and Ferreira, R. A. (2025). GenBucket: Source-Code Repository. `https://github.com/MiltonBaze/Genbucket`.

Ben-Sasson, H. and Greenberg, R. (2023). 38TB of data accidentally exposed by Microsoft AI researchers. `https://www.wiz.io/blog/38-terabytes-`

of-private-data-accidentally-exposed-by-microsoft-ai-researchers/.

Bouchet, M. et al. (2020). Block Public Access: Trust Safety Verification of Access Control Policies. In *Proc. of ACM FSE*, pages p.281–291.

BR-Office (2016). Verificador Ortografico pt-br. https://sourceforge.net/projects/verificador/.

Cable, J., Gregory, D., Izhikevich, L., and Durumeric, Z. (2021). Stratosphere: Finding Vulnerable Cloud Storage Buckets. In *Proc. of RAID 2021*, page 399–411.

Cisoadvisor (2023). Securitas Expõe Três Terabytes de Dados de Aeroportos na Colômbia e Peru. https://www.cisoadvisor.com.br/securitas-expoe-3-tb-de-dados-de-aeroportos-na-colombia-e-peru/.

Continella, A. et al. (2018). There's a Hole in that Bucket! A Large-scale Analysis of Misconfigured S3 Buckets. In *Proc. of ACSAC*, pages p.702–711.

Donda, D. (2018). https://github.com/danieldonda/wordlist/.

Eldad, D. (2023). The Danger of Publicly Exposed S3 Buckets. https://www.rubrik.com/blog/technology/23/3/the-data-on-the-danger-of-publicly-exposed-s3-buckets/.

EleutherAI (2023). GPT-Neo. https://www.eleuther.ai/artifacts/gpt-neo/.

Google (2025). Google Cloud, Produtos de Armazenamento Online do Google Cloud. https://cloud.google.com/products/storage?hl=pt-BR.

Grayhatwarfare (2018). Search Public Buckets. https://buckets.grayhatwarfare.com/.

Houdt, G. V. et al. (2020). A Review on the Long Short-Term Memory Model. In *Artificial Intelligence Review*, volume 53, page 5929–5955. Artif Intell Rev 53.

Kelley, P. et al. (2012). Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In *Proc. of IEEE Security and Privacy 2012*.

Mari, A. (2020). Brazilian firm exposes personal details of thousands of soccer fans. https://www.zdnet.com/article/brazilian-firm-exposes-personal-details-of-thousands-of-soccer-fans/.

Microsoft (2023). Recomendações de Segurança para o Armazenamento de Blobs. https://learn.microsoft.com/pt-br/azure/storage/blobs/security-recommendations.

Mushtaq, F. (2025). https://www.pureprivacy.com/blog/dark-web-monitoring/hipshipper-data-breach. https://www.pureprivacy.com/blog/dark-web-monitoring/hipshipper-data-breach/.

NordSecurity (2023). Passwords List/. https://nordpass.com/most-common-passwords-list/.

Ocean, D. (2025). Highly Scalable and Affordable Object Storage. https://www.digitalocean.com/products/spaces.

Paszke, A. et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Proc of NeurIPS 2019*, volume 32.

ProjectDiscovery (2024). Nuclei - fast and customizable vulnerability scanner based on templates. https://github.com/projectdiscovery/nuclei.

Qualys, Inc. (2025). Qualys Web Application Scanning (WAS). https://www.qualys.com/apps/web-application-scanning/.

Research, G. V. (2025). Global Cloud Computing Market Size & Outlook, 2024-2030. https://www.grandviewresearch.com/horizon/outlook/cloud-computing-market-size/global.

Surfshark (2022). Brasil é o 6º País com mais Vazamentos de Dados no Planeta, Aponta Levantamento. https://www.istoedinheiro.com.br/seguranca-de-dados-brasil-e-o-6o-pais-com-mais-vazamentos-diz-pesquisa/.

Surribas, N. (2006). Wapiti: a Free and Open-Source web-application vulnerability scanner in Python. https://wapiti-scanner.github.io/.

Tenable (2024). 2024 Cloud Security Outlook: Navigating Barriers and Setting Priorities. https://www.tenable.com/cyber-exposure/2024-cloud-security-outlook.

Tenable, Inc. (2024). Nessus professional vulnerability scanner. https://www.tenable.com/products/nessus.

Vaswani, A. et al. (2017). Attention is all you need. In *Proc. of NIPS 2017*.

VirusTotal (2022). VirusTotal Intelligence. https://www.virustotal.com/gui/home/upload.

Weaver, K. (2017). This is a demo of setting up an Amazon Web Service (AWS) S3 bucket and uploading a file with Python. /. https://github.com/keithweaver/python-aws-s3/.

Weir, M. et al. (2009). Password Cracking Using Probabilistic Context-Free Grammars. In *Proc. of IEEE Security and Privacy 2009*.

Westervelt, R. (2013). Amazon S3 Users Exposing Sensitive Data, Study Finds. https://www.crn.com/news/security/240151857/amazon-s3-users-exposing-sensitive-data-study-finds.

Willis (2013). There's a Hole in 1.951 Amazon S3 Buckets. https://www.rapid7.com/blog/post/2013/03/27/open-s3-buckets/.

Wolf, T. et al. (2020). Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 EMNLP*, pages 38–45.

Wood, R. (2011). Analysing Amazon's Buckets. https://digi.ninja/blog/analysing_amazons_buckets.php.

Yadmani, S. E. et al. (2025). The File That Contained the Keys Has Been Removed: An Empirical Analysis of Secret Leaks in Cloud Buckets and Responsible Disclosure Outcomes. In *Proc. of IEEE Security and Privacy 2025*, page 9–9.