VulnSyncAI: PLN e LLMs para Construção e Atualização Contínua de Datasets de Vulnerabilidades

Douglas Rodrigues Fideles[1], **Douglas Paim Lautert**[1], **Diego Luis Kreutz**[1], **Silvio Ereno Quincozes**[1]

[1]Universidade Federal do Pampa (UNIPAMPA)

{douglasfideles,douglaslautert}aluno.@unipampa.edu.br

{diegokreutz,silvioquincozes}@unipampa.edu.br

Abstract. *The construction and maintenance of up-to-date vulnerability datasets face challenges such as lack of standardization and the need for automation. In this work, we present VulnSyncAI, a modular tool that uses NLP (Natural Language Processing) and LLMs (Large Language Models) to correlate information from multiple sources, ensuring updated and relevant datasets. VulnSyncAI enhances the effectiveness of AI models in threat detection by automating processes and increasing efficiency in creating representative datasets.*

Resumo. *A construção e manutenção de datasets atualizados de vulnerabilidades enfrentam desafios como falta de padronização e necessidade de automação. Neste trabalho, apresentamos a VulnSyncAI, uma ferramenta modular que utiliza PLN e LLMs para correlacionar informações de múltiplas fontes, garantindo datasets atualizados e relevantes. A VulnSyncAI melhora a eficácia de modelos de IA na detecção de ameaças, automatizando processos e aumentando a eficiência na criação de datasets representativos.*

1. Introdução

A detecção de vulnerabilidades é essencial para proteger aplicativos de software contra ameaças à segurança. A detecção automatizada permite que organizações respondam rapidamente a riscos potenciais, especialmente diante do aumento contínuo de vulnerabilidades. Em 2022, foram relatadas 25.227 vulnerabilidades, e 5.910 foram identificadas apenas nos primeiros três meses de 2023 [Guo and Bettaieb 2024]. Conjuntos de dados abrangentes e atualizados são fundamentais para pesquisadores e profissionais de segurança, pois permitem a identificação de tendências, o desenvolvimento de estratégias defensivas e o treinamento de modelos de aprendizado de máquina [Hoang et al. 2020].

No entanto, a construção e manutenção de datasets de alta qualidade apresentam desafios significativos [Guo et al. 2024, Whang et al. 2023, Hu et al. 2021]. Um dos principais problemas é a dificuldade de manter os datasets atualizados, o que é crucial para a eficácia dos modelos de IA. A natureza dinâmica das vulnerabilidades, que surgem e evoluem rapidamente, torna a detecção de ameaças extremamente difícil. Abordagens existentes frequentemente dependem de grandes volumes de dados rotulados, que são escassos, caros e demorados para serem obtidos. Ademais, a qualidade dos dados em repositórios públicos é variável, com inconsistências, informações incompletas e atrasos na atualização [Croft et al. 2023]. Esses fatores resultam em modelos de IA treinados com dados desatualizados, comprometendo sua capacidade de detectar ameaças emergentes.

O principal desafio é a criação e o gerenciamento de *datasets* atualizados, que permitam a análise eficiente das limitações de protocolos e sistemas. Para lidar com a natureza dinâmica das vulnerabilidades e a demanda por dados em tempo real, é necessária uma solução que permita a coleta automatizada e massiva de metadados, a automação do processo de classificação e a implementação de mecanismos de atualização contínua. Estudos recentes demonstram que a obtenção de dados atualizados e representativos é uma tarefa complexa e custosa [Guo et al. 2024, Whang et al. 2023]. A obsolescência dos *datasets* disponíveis foi identificada como um dos principais problemas e desafios em diferentes domínios. A criação e atualização contínua de *datasets* representativos e atualizados é fundamental para o sucesso de qualquer processo de análise de vulnerabilidades em protocolos ou sistemas, bem como a detecção e prevenção de ataques.

Atualmente, existem diferentes fontes de dados sobre vulnerabilidades de protocolos e sistemas, como o U.S. *National Vulnerability Database* (NVD)[1], o banco de dados dinamarquês Secunia[2], o SecurityFocus[3], a Plataforma Chinesa de Compartilhamento de Vulnerabilidades de Segurança da Informação (CNVD)[4], o Banco de Dados de Vulnerabilidades de Segurança da Informação da China (CNNVD)[5], e o NSFocus[6]. Essas fontes são normalmente disponibilizadas de diferentes formas e nem sempre utilizam o mesmo formato de dados de dados, o que dificulta o processo de coleta de dados e construção de conjuntos de dados de vulnerabilidades.

A construção de conjuntos de dados de vulnerabilidades também enfrenta desafios como a confiabilidade das fontes de dados, a falta de informações em múltiplas fontes, a seleção de características relevantes e a granularidade dos dados [Lin et al. 2022]. A discrepância entre diferentes bases de dados, como NVD e SecurityFocus, pode levar a redundâncias e erros, exigindo esforços manuais para verificação. Além disso, a ausência de valores em dados multivariados e a dificuldade em selecionar características adequadas, como assinaturas de patches, impactam a eficácia dos modelos de detecção. A granularidade dos dados também é crucial, com foco em fragmentos de código vulneráveis, em vez de apenas funções, para melhorar a precisão. O tamanho do conjunto de dados é outro desafio, pois conjuntos pequenos podem não cobrir tipos suficientes de vulnerabilidades, enquanto conjuntos grandes podem levar ao esquecimento catastrófico em modelos de IA.

Para abordar esses desafios, este trabalho apresenta VulnSyncAI, uma ferramenta modular e extensível que utiliza técnicas de Processamento de Linguagem Natural (PLN) e grandes modelos de linguagem (LLMs) para correlacionar informações de múltiplas fontes e protocolos. A VulnSyncAI coleta, organiza e categoriza dados, além de incorporar mecanismos de atualização contínua, garantindo que os datasets reflitam as últimas vulnerabilidades reportadas nas bases online. Isso permite que modelos de IA sejam treinados com dados atualizados e relevantes, melhorando sua eficácia na detecção de ameaças modernas. A ferramenta também integra técnicas de automação para a rotulagem de dados, reduzindo a dependência de processos manuais e aumentando a eficiência na criação de datasets representativos.

[1] `https://nvd.nist.gov/`
[2] `http://secunia.com/`
[3] `http://www.securityfocus.com/vulnerabilities`
[4] `http://www.cnvd.org.cn/`
[5] `http://www.cnnvd.org.cn/`
[6] `http://www.nsfocus.net/index.php`

2. Trabalhos Relacionados

A Tabela 1 destaca as principais características das abordagens existentes e como a VulnSyncAI se diferencia ao oferecer uma solução agnóstica a domínios, integrando múltiplas fontes (NVD, Vulners) e processamento automatizado com LLMs para categorização. Diferentemente de ferramentas como *ADBuilder* [Vilanova et al. 2022] e *AMGenerator* [Rocha et al. 2023], que são voltadas especificamente para *Android*, a VulnSyncAI é independente de domínio. Enquanto soluções como o *SentinelData-Downloader* [Sebastianelli et al. 2021] geram dados estáticos, a VulnSyncAI implementa atualização contínua, essencial para o gerenciamento de vulnerabilidades dinâmicas. Além disso, sua arquitetura modular permite extensão por meio de novos conectores (e.g., CNVD) e modelos (e.g., *SLMs* locais), superando a rigidez de soluções como o *VIETTool*, que não suportam adaptação a novos formatos.

Ferramenta/Trabalhos	Usa IA?	Domínios	Extensível?	Fontes	Foco
ADBuilder [Vilanova et al. 2022]	Sim	Malware Android	Sim	Múltiplas	Construção automatizada de datasets para detecção de malwares Android
AMGenerator [Rocha et al. 2023]	Sim	Malware Android	Sim	Múltiplas	Evolução do ADBuilder. Metadados e datasets Android
VIETTool [Zhang et al. 2023]	Sim	Vulnerabilidades (Texto), Protocolos	Não	Múltiplas	Extração de Entidades (NER)
VulDeePecker [Zou et al. 2021]	Sim	Vulnerabilidades (Código)	Não	Múltiplas	Detecção de Vulnerabilidades (nível de código)
SentinelDataDownloaderTool [Sebastianelli et al. 2021]	Sim	Imagens de Satélite	Não	Única	Construção de datasets com imagens de satélite
Software Metrics and Security Vulnerabilities [Alves et al. 2016]	Não	Vulnerabilidades (Métricas de Software)	Não	Única	Construção de dataset com métricas de software
VulnSyncAI (este trabalho)	**Sim**	**Vulnerabilidades (Agnóstico)**	**Sim**	**Múltiplas**	**Construção e Atualização Contínua de Datasets de Vulnerabilidades com IA**

Tabela 1. Trabalhos Relacionados

A literatura apresenta diversas abordagens para a construção de *datasets* de vulnerabilidades e o uso de *IA* em segurança de software. Algumas ferramentas focam na criação automatizada de *datasets* para detecção de *malwares* Android [Rocha et al. 2023, Vilanova et al. 2022], enquanto o [Sebastianelli et al. 2021] constrói *datasets* a partir de imagens de satélite. Entretanto, essas abordagens geralmente se concentram em domínios ou tarefas específicas, limitando sua aplicabilidade em contextos mais amplos.

Comparado a abordagens baseadas em dados sintéticos ou pré-existentes, o foco do VulnSyncAI em correlação em vários formatos e padronização inteligente (via normalização e LLMs) garante *datasets* atualizados e estruturalmente consistentes. A integração de LLMs para categorização (e.g., *Gemini*, *LLaMA*) vai além da extração de entidades, identificando relações complexas como causa-impacto, um diferencial ante ferramentas como *Software Metrics* [Alves et al. 2016], limitadas a métricas estáticas.

3. Arquitetura

A arquitetura da ferramenta VulnSyncAI, ilustrada na Figura 1, é organizada em quatro módulos principais, seguindo um fluxo de processamento sequencial e modular: **Coleta de Dados**, **Processamento**, **Categorização** e **Geração do Dataset**. Cada módulo possui a garantia da separação de preocupações o que facilita a extensibilidade.

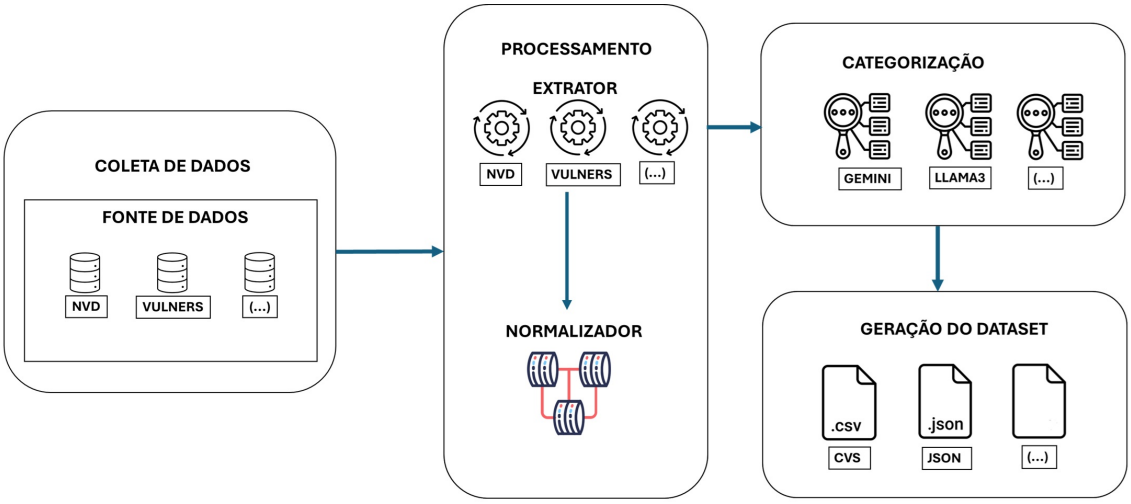

Figura 1. Overview da Arquitetura da VulnSyncAI

O módulo de **Coleta de Dados** atua como interface com fontes externas de vulnerabilidades, como NVD, Vulners e CNVD. Por meio de conectores especializados, ele estabelece comunicação com APIs, realiza o download de dados brutos (metadados, descrições textuais) conforme parâmetros definidos pelo usuário.

No módulo de **Processamento**, os dados coletados são normalizados e padronizados, removem-se as duplicatas com base em identificadores únicos (e.g., CVE-ID), unificam-se formatos, ajustando padrões de dados e criticidade. O resultado é um dataset intermediário padronizado, independente da origem dos dados.

O módulo de **Categorização** utiliza LLMs via API ou SLMs locais (e.g Gemini Pro e LLaMA 3) para classificar e enriquecer vulnerabilidades. Ao categorizar são extaídas informações estruturadas das descrições textuais incluindo, classificação CWE, impacto, causa raiz, fornecedor e recomendações de mitigação. Cada saída é rastreável.

O módulo de **Geração do Dataset** unifica os resultados da categorização em um dataset consolidado, formata a saída conforme escolhido (e.g., CSV, JSON), valida a consistência dos dados. Além disso, atualiza bases históricas para permitir versionamento e monitoramento temporal de vulnerabilidades.

3.1. Extensibilidade

A arquitetura modular do *VulnSyncAI* é extensível, permitindo a integração de novas fontes de dados, modelos de *IA* e formatos de saída sem a necessidade de alterar o código principal. Essa abordagem garante adaptabilidade a diversas necessidades e fontes de informação, mantendo a ferramenta flexível, eficiente e pronta para expansão.

No módulo de **Coleta de Dados**, para adicionar uma nova fonte de dados (e.g., ”MinhaNovaFonte”), é necessário implementar um novo componente de conexão. Esse componente deve ser uma classe *Python*, criada no diretório `data_sources/`, e seguir a interface `DataSourceBase` (definida em `data_sources/data_source.py`). A implementação exige dois métodos principais: `collect_data(search_params)`, responsável por buscar os dados, e `normalize_data(vulnerability)`, que realiza uma padronização inicial, se necessária. Após a implementação, o componente é

registrado no arquivo `config.yaml`.

No módulo de **Processamento**, para cada nova fonte de dados, um componente de extração deve ser criado. Esse componente, também herdeiro de `DataSourceBase`, implementa o método `normalize_data()` para extrair informações relevantes, como descrição, ID, entre outros. O registro é feito por meio da função `load_data_sources` em `data_sources/load_data_source.py`.

Para adicionar um novo LLM ou SLM, basta configurar o arquivo `config.yaml`. Para LLMs, são configuradas chaves de API, URLs e outros parâmetros, enquanto para *SLMs*, é necessário especificar o modelo desejado.

Por fim, a **Geração do Dataset** também é extensível. Para isso, é necessário criar um novo arquivo *Python* (e.g., `novo_exportador.py`) no diretório `output/`, com uma classe (e.g., `NovoSaida`) que herde de `DataExporterBase` e implemente o método `export(data, filename)`. Esse método recebe as vulnerabilidades e o nome do arquivo de saída, escrevendo os dados no formato desejado.

3.2. Implementação e Utilização

A ferramenta *VulnBuilderAI* foi desenvolvida em *Python* (versão 3.10) para processar dados, realizar requisições web e integrar modelos de linguagem (*LLMs* e *SLMs*). Utiliza bibliotecas como `requests` (versão 2.30.0), `aiohttp` (versão 3.8.5), `json`, `csv`, `google-generativeai` (versão 0.3.0), `openai` (versão 0.27.4), `pytorch` (versão 2.0.1) e `transformers` (versão 4.29.0), entre outras, para garantir flexibilidade e eficiência. O código-fonte está disponível no repositório [7]. Um exemplo de execução da ferramenta e seus respectivos parâmetros de entrada é apresentado a seguir:

```
python src/main.py --provider '[PROVEDOR(S) LLM]'
--data-source '[FONTE(S) DE DADOS]' --search-params
'[PARAMETRO(S) DE BUSCA]' --export-format [FORMATO(S) DE
SAIDA] --output-file [ARQUIVO DE SAIDA]
```

4. Resultados Experimentais e Discussão

A ferramenta *VulnSyncAI* foi avaliada quanto à sua capacidade de coletar, normalizar e categorizar vulnerabilidades, com foco em quatro estudos de caso: vulnerabilidades do protocolo MQTT, vulnerabilidades em navegadores web, vulnerabilidades em sistemas de Veículos Aéreos Não Tripulados (UAVs) e vulnerabilidades no *Data Distribution Service (DDS)*. Os dois últimos, devido à sua extensão e para manter o foco nos resultados principais, são apresentados em detalhes no Apêndice.

O objetivo desta avaliação é analisar o desempenho da ferramenta ao extrair e classificar informações relevantes a partir de descrições de vulnerabilidades, e não a detecção de vulnerabilidades em si. Utilizamos como fontes de dados o *National Vulnerability Database* (NVD) e a API do *Vulners*.

4.1. Vulnerabilidades no Protocolo MQTT

Foram utilizadas implementações populares do MQTT como parâmetros de busca: Eclipse Mosquitto, EMQX, VerneMQ, RabbitMQ e HiveMQ [Longo et al. 2022]. A coleta de dados foi realizada a partir do National Vulnerability Database (NVD) e da API do

[7] `https://github.com/datasets-community/VulnSyncAI`

Vulners, e a categorização foi executada utilizando três configurações de LLMs: Gemini Pro 1.5 (via API), LLaMA 3 (local) e Deepseek (API).

Na fase de extração de dados, foram coletadas 88 vulnerabilidades únicas quando a fonte de dados é o NVD, e 82 quando a fonte é o Vulners, abrangendo o período de 2014 a 2025. A diferença no total de vulnerabilidades coletadas entre as fontes NVD e Vulners destaca a importância de utilizar múltiplas fontes para uma cobertura mais abrangente. O processo de normalização padronizou os dados, independentemente da fonte, preparando-os para a categorização. É importante notar que os totais de vulnerabilidades por fornecedor são idênticos entre os modelos. Isso ocorre porque a extração do fornecedor, nesta etapa, não é realizada pelo LLM, mas sim durante a fase de normalização, utilizando os parâmetros de busca.

Fornecedor	**Distribuição**	**CRÍTICA**	**ALTA**	**MÉDIA**	**BAIXA**	**NENHUM**
RabbitMQ	78	9	15	3	22	14
Eclipse Mosquitto	44	2	21	0	13	0
EMQX	23	1	4	0	15	2
VerneMQ	20	0	10	0	0	4
HiveMQ	5	0	0	0	3	1
Desconhecido	0	0	0	0	0	0

Tabela 2. Vulnerabilidades Por Fabricante e Criticidade (Fontes: NVD + Vulners)

A Tabela 2 ilustra a distribuição das vulnerabilidades por fornecedor e criticidade, combinando os dados das duas fontes. O RabbitMQ lidera com um número expressivo de 78 vulnerabilidades. A alta concentração em RabbitMQ e Eclipse Mosquitto pode estar associada à popularidade e ampla utilização desses *brokers*.

Foi observada uma diferença significativa nos resultados da categorização CWE atribuída pelos diferentes LLMs, mesmo utilizando a mesma fonte de dados. O LLaMA 3, na configuração local, apresentou uma alta taxa de ”Desconhecido”(82,7% no NVD e 53,8% no Vulners). Isso pode ser atribuído a limitações do modelo local, necessidade de *fine-tuning* ou ao formato da descrição. O Gemini Pro 1.5, acessado via API, obteve resultados melhores, mas ainda com um número considerável de ”Desconhecido”e erros. O Deepseek, ao categorizar com a fonte de dados NVD, identificou as categorias mais frequentes como: CWE-400 (9), CWE-798 (6), CWE-79 (6), CWE-502 (5) e CWE-20 (5). Já com a fonte de dados Vulners, o Deepseek obteve: CWE-400 (13), CWE-937 (12), CWE-120 (7), CWE-787 (7) e CWE-200 (6).

A fonte de dados (NVD vs. Vulners) também influencia a categorização, mesmo com o mesmo LLM. Isso sugere que as diferenças nas descrições das vulnerabilidades entre as fontes impactam a capacidade dos LLMs de extrair informações relevantes.

Apesar da variação, algumas categorias CWE aparecem com frequência em vários modelos e fontes, como CWE-200 (Exposição de Informações), CWE-400 (Consumo de Recursos Não Controlado), CWE-79 (*Cross-site Scripting*) e CWE-20 (Validação de Entrada Inadequada). Isso indica que essas são vulnerabilidades comuns em implementações do MQTT.

O Gemini, com a fonte de dados NVD, obteve a maioria das categorias mais frequentes, sendo CWE-200, com 10 ocorrências. Em contrapartida, o Deepseek obteve a maioria com a fonte de dados Vulners, sendo CWE-400, com 13 ocorrências.

4.2. Vulnerabilidades em Navegadores Web

Neste estudo de caso, aplicamos a ferramenta para coletar e analisar vulnerabilidades relacionadas a navegadores web populares, de acordo com as estatísticas de acesso do W3Schools [W3Schools 2024]. A coleta de dados foi realizada a partir da fonte NVD, e a categorização foi executada utilizando duas configurações de LLM: Gemini Pro 1.5 (via API) e DeepHermes-3-Llama-3-8B-Preview3 (local). O objetivo é demonstrar a versatilidade da ferramenta ao lidar com domínios diferentes.

Fornecedor	Distribuição	CRÍTICA	ALTA	MÉDIA	BAIXA
Mozilla Firefox Browser	196	2	14	0	14
Microsoft Edge Browser	122	0	1	0	2
Opera Browser	98	3	2	0	1
Google Chrome Browser	84	3	22	1	36
Desconhecido	0	0	0	0	0

Tabela 3. Vulnerabilidades Por Fabricante e criticidade

A coleta de dados resultou em um total de 508 vulnerabilidades relacionadas aos navegadores mencionados. Como apresentado na Figura 3, o Mozilla Firefox apresenta o maior número de vulnerabilidades identificadas, seguido pelo Microsoft Edge (122) e Google Chrome (98).

A análise das vulnerabilidades de navegadores web revela padrões importantes e destaca áreas de foco para a segurança. As categorias mais frequentes foram CWE-119 (*Improper Restriction of Operations within the Bounds of a Memory Buffer*) com 117 ocorrências, CWE-787 (*Out-of-bounds Write*) com 95, CWE-200 (*Information Exposure*) com 53, CWE-79 (*Cross-site Scripting*) com 53 e CWE-20 (*Improper Input Validation*) com 33. Um número significativo de vulnerabilidades (157) não foi categorizado pelo modelo, sendo classificado como ”Desconhecido”.

Na categorização CWE, observamos que o DeepHermes-3-Llama-3-8B-Preview (na configuração local) classificou como ”Desconhecido” 420 de 508 vulnerabilidades, ou seja, 82,7%. Isso indica que, na grande maioria dos casos, o modelo não conseguiu identificar a vulnerabilidade. Esse resultado contrasta com o Gemini Pro 1.5, que, embora também tenha apresentado resultados ”Desconhecido”, categorizou a maioria das vulnerabilidades.

5. Demonstração e Considerações Finais

Demonstração. A ferramenta será apresentada por meio de uma máquina virtual hospedada na Google Cloud. As funcionalidades serão demonstradas seguindo a seção de experimentos da documentação da ferramenta no GitHub [8].

Conclusão. A *VulnSyncAI* demonstrou eficácia na construção dinâmica de *datasets*, com três contribuições principais: arquitetura modular para coleta multiformato, categorização semântica via LLMs e mecanismos de atualização contínua. Entre as limitações, destacam-se taxas elevadas de ”Desconhecido” em SLMs locais (25–53%), associadas à complexidade dos *prompts* e à natureza não estruturada dos dados.

[8] `https://github.com/datasets-community/VulnSyncAI`

Trabalhos futuros. Futuros trabalhos devem explorar o *fine-tuning* de LLMs para CWEs e o desenvolvimento de interfaces visuais para usuários não técnicos. A otimização dos *prompts*, a implementação de técnicas de pós-processamento para "limpar"as saídas dos LLMs e a extração de informações relevantes são direções promissoras para aumentar a cobertura e a precisão da ferramenta.

Referências

Alves, H., Fonseca, B., and Antunes, N. (2016). Software metrics and security vulnerabilities: Dataset and exploratory study. In *2016 12th European Dependable Computing Conference (EDCC)*, pages 37–44.

Croft, R., Babar, M. A., and Kholoosi, M. M. (2023). Data quality for software vulnerability datasets. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 121–133. IEEE.

Guo, Y. and Bettaieb, S. (2024). An investigation of quality issues in vulnerability detection datasets. In *arXiv preprint arXiv:2410.06030*.

Guo, Y., Bettaieb, S., and Casino, F. (2024). A comprehensive analysis on software vulnerability detection datasets: trends, challenges, and road ahead. *International Journal of Information Security*, 23(5):3311–3327.

Hoang, T., Kang, H. J., Lo, D., and Lawall, J. (2020). Cc2vec: distributed representations of code changes. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 518–529, New York, NY, USA. Association for Computing Machinery.

Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. (2021). Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*.

Lin, Y., Li, Y., Gu, M., Sun, H., Yue, Q., Hu, J., Cao, C., and Zhang, Y. (2022). Vulnerability dataset construction methods applied to vulnerability detection: A survey. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 141–146.

Longo, E., Redondi, A. E. C., Cesana, M., and Manzoni, P. (2022). Border: A benchmarking framework for distributed mqtt brokers. *IEEE Internet of Things Journal*, 9(18):17728–17740.

Rocha, V., Assolin, J., Bragança, H., Kreutz, D., and Feitosa, E. (2023). Amgenerator e amexplorer: Geração de metadados e construção de datasets android. In *Anais Estendidos do XXIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 41–48, Porto Alegre, RS, Brasil. SBC.

Sebastianelli, A., Del Rosso, M. P., and Ullo, S. L. (2021). Automatic dataset builder for machine learning applications to satellite imagery. *SoftwareX*, 15:100739.

Vilanova, L., Kreutz, D., Assolin, J., Quincozes, V., Miers, C., Mansilha, R., and Feitosa, E. (2022). Adbuilder: uma ferramenta de construção de datasets para detecção de malwares android. In *Anais Estendidos do XXII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 143–150, Porto Alegre, RS, Brasil. SBC.

W3Schools (2024). Browser statistics. Available: https://www.w3schools.com/browsers/. Accessed on 2025-01-27.

Whang, S. E., Roh, Y., Song, H., and Lee, J.-G. (2023). Data collection and quality challenges in deep learning: A data-centric ai perspective. *The VLDB Journal*, 32(4):791–813.

Zhang, S., Zhang, M., and Zhao, L. (2023). Viet: A tool for extracting essential information from vulnerability descriptions for cvss evaluation. In *Data and Applications Security and Privacy XXXVII: 37th Annual IFIP WG 11.3 Conference, DBSec 2023, Sophia-Antipolis, France, July 19–21, 2023, Proceedings*, page 386–403, Berlin, Heidelberg. Springer-Verlag.

Zou, D., Wang, S., Xu, S., Li, Z., and Jin, H. (2021). μvuldeepecker: A deep learning-based system for multiclass vulnerability detection. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2224–2236.

Apêndice: Ambiente

Para rodar o estudo de caso foram utilizados a configuração SLM Llama3 e Gemini e Deepseek para API:

- LLMs do Gemini e DeepSeek: Computador local com Intel Core i7-7700HQ, 16GB RAM, Windows 10.
- SLM do Llama3 (DeepHermes-3-Llama-3-8B-Preview3): Máquina virtual na Google Cloud com 12 vCPUs, 40GB RAM, Ubuntu 20.04.

O código-fonte completo do VulnBuilderAI, juntamente com instruções detalhadas de instalação, configuração e uso, está disponível no seguinte repositório GitHub [9].

Apêndice: Caso de uso DDS - *Data Distribution Service*

A imagem 2 apresenta a distribuição das vulnerabilidades por fornecedor, obtidas a partir da fonte NVD.

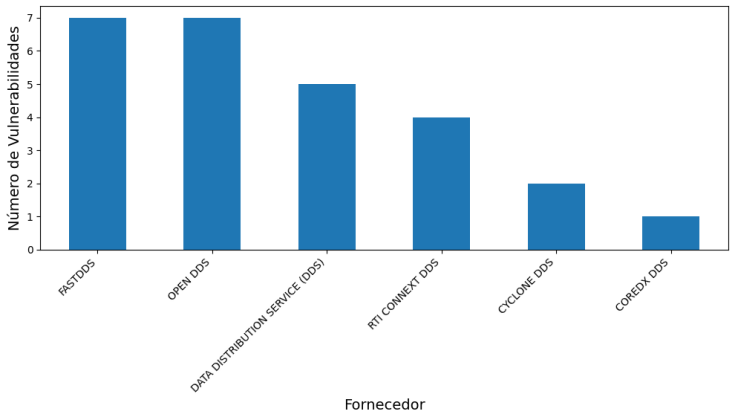

Figura 2. Vulnerabilidades por Fabricante

[9] https://github.com/datasets-community/VulnSyncAI

Gemini (API)

A imagem 3 apresenta as cinco categorias CWE (Common Weakness Enumeration) mais frequentemente atribuídas pelo modelo Gemini (API) às vulnerabilidades únicas encontradas, utilizando a fonte de dados NVD para o dataset DDS.

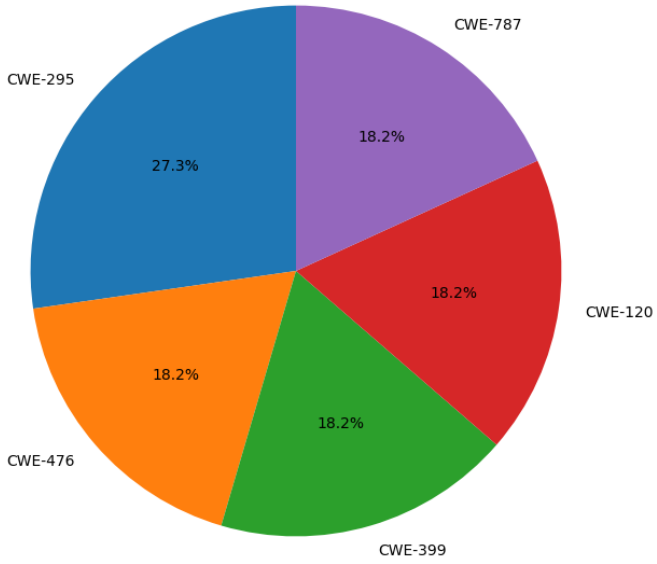

Figura 3. Categorização Gemini CWE Mais Frequentes

LLAMA3 (Local)

As figuras a seguir detalham os resultados da categorização utilizando o modelo LLaMA 3 (executado localmente na máquina virtual) e a fonte de dados NVD, para o dataset de vulnerabilidades de UAVs.

A figura 4 apresenta as cinco categorias CWE (Common Weakness Enumeration) mais frequentemente atribuídas pelo modelo Llama3 (Local) às vulnerabilidades únicas encontradas, utilizando a fonte de dados NVD para o dataset DDS.

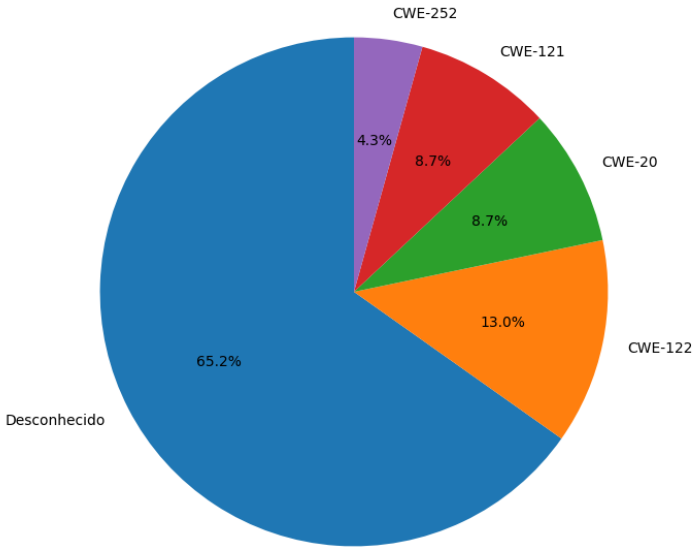

Figura 4. Categorização Llama3 CWE Mais Frequentes

6. Apêndice: caso de uso UAV

Nesta figura 5 apresenta a distribuição das vulnerabilidades por fornecedor, obtidas a partir da fonte NVD.

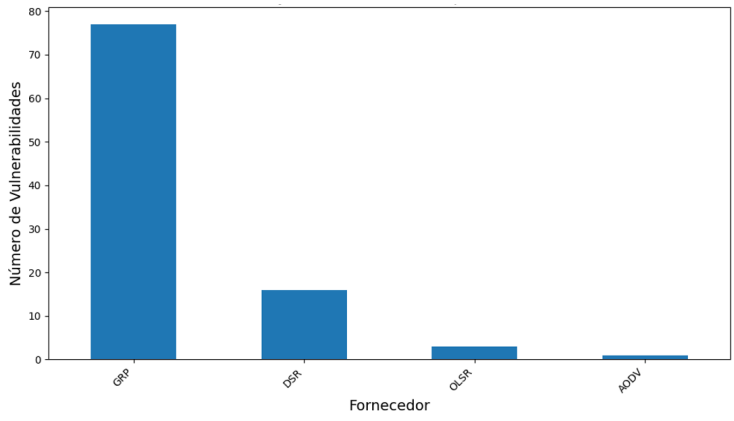

Figura 5. Vulnerabilidades por Fabricante

Gemini (API)

A figura 6 apresenta as cinco categorias CWE (Common Weakness Enumeration) mais frequentemente atribuídas pelo modelo Gemini Pro 1.5 (API) às vulnerabilidades únicas encontradas, utilizando a fonte de dados NVD para o dataset UAV.

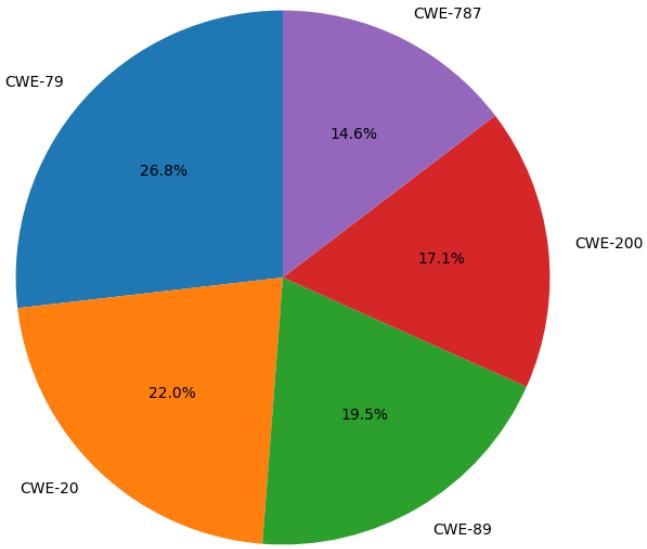

Figura 6. Categorização Gemini CWE Mais Frequentes

Llama3 (Local)

A figura 7 apresenta as cinco categorias CWE (Common Weakness Enumeration) mais frequentemente atribuídas pelo modelo Llama3 (local) às vulnerabilidades únicas encontradas, utilizando a fonte de dados NVD para o dataset UAV.

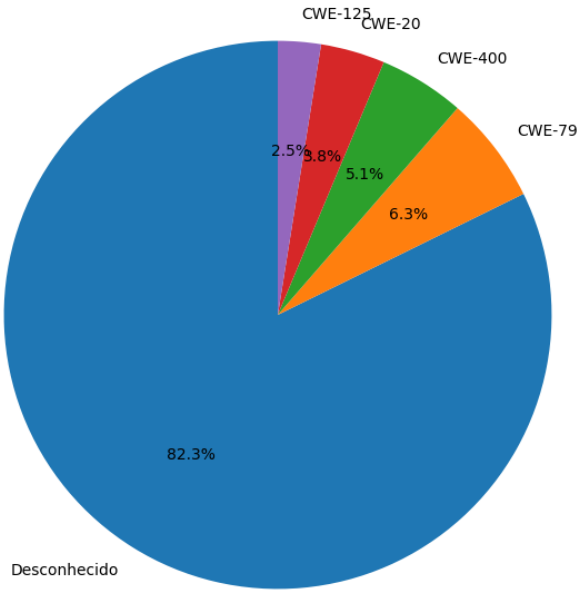

Figura 7. Categorização Llama3 CWE Mais Frequentes