OpenVAS Agent: Melhoria de Experiência do Usuário no OpenVAS utilizando LLM

Raphael Alves de Lima Soares[1], Lucas Nogueira Barroso[1], Antônio João Gonçalves de Azambuja[2], Jean Phelipe de Oliveira Lima[2], Leonardo Barbosa Oliveira[3], Anderson da Silva Soares[1]

[1] Instituto de Informática — Universidade Federal de Goiás (UFG) — Goiás — Brasil
[2] Instituto Tecnológico de Aeronáutica (ITA) — São Paulo — Brasil
[3] Universidade Federal de Minas Gerais (UFMG) — Minas Gerais — Brasil
raphael_alves@discente.ufg.br, lucas.nogueira@discente.ufg.br

Abstract. *Vulnerability analysis is a standard technique within the context of cybersecurity; however, tools such as OpenVAS still present operational barriers for less experienced users. This work proposes a solution that integrates OpenVAS with an Artificial Intelligence (AI) agent based on Large Language Models (LLMs), allowing for scan configuration and report interpretation using accessible commands. The architecture employs the LangChain framework and the ReAct approach to automate tasks and transform technical data into understandable insights. System validation characterized the operational effort required by each interface: creating a scan requires 11 clicks in the graphical interface or 7 chained commands in the command line, whereas the agent conducts a guided six-turn dialogue in which each required parameter is explicitly elicited, removing the need for prior knowledge of the tool's entity taxonomy. Thus, the tool optimizes access to complex offensive security processes.*

Resumo. *A análise de vulnerabilidades é uma técnica utilizada no contexto da segurança cibernética; no entanto, ferramentas como o OpenVAS ainda apresentam barreiras operacionais para usuários menos experientes. Este trabalho propõe uma solução que integra o OpenVAS a um agente de Inteligência Artificial (IA) baseado em Large Language Models (LLMs), permitindo configurar varreduras e interpretar relatórios usando comandos acessíveis. A arquitetura emprega o framework LangChain e a abordagem ReAct para automatizar tasks e transformar dados técnicos em insights compreensíveis. A validação do sistema caracterizou o esforço operacional exigido por cada interface: a criação de uma varredura demanda 11 cliques na interface gráfica ou 7 comandos encadeados na linha de comando, enquanto o agente conduz um diálogo guiado de seis turnos no qual cada parâmetro necessário é solicitado explicitamente, dispensando o conhecimento prévio da taxonomia de entidades da ferramenta. Assim, a ferramenta otimiza o acesso a processos complexos de segurança ofensiva.*

1. Introdução

A análise de vulnerabilidades é apresentada como uma etapa central para a proteção de redes de computadores em um cenário contemporâneo, onde ataques cibernéticos tornam-se cada vez mais sofisticados e automatizados. Historicamente, a identificação antecipada

e a mitigação estratégica de riscos de segurança passaram por diversos estágios evolutivos, culminando na adoção de ferramentas especializadas e robustas. Dentre estas, *OpenVAS* (*Open Vulnerability Assessment System*) [Sharma et al. 2024] figura como um dos pilares fundamentais no arsenal de cibersegurança global [Odun-Ayo et al. 2024].

Entretanto, apesar de sua robustez técnica e ampla aceitação pela comunidade de segurança, o *OpenVAS* impõe desafios operacionais significativos que funcionam como barreiras de entrada. Estes desafios manifestam-se primordialmente em três frentes:

- **Configuração e Instalação**: O ecossistema GVM (*Greenbone Vulnerability Management*) exige uma instalação minuciosa de serviços, cujas dependências e comunicações via sockets Unix podem ser complexas para administradores de sistemas generalistas.
- **Densidade de Relatórios**: Um escaneamento de rede típico pode gerar relatórios com milhares de entradas, contendo descrições técnicas densas, pontuações CVSS (*Common Vulnerability Scoring System*) e referências externas que exigem um esforço hercúleo de triagem manual.
- **Dificuldade de Interpretação**: A extração rápida de *insights* acionáveis para a tomada de decisão executiva é frequentemente prejudicada pela poluição de dados técnica, dificultando a distinção entre vulnerabilidades críticas e falsos positivos em ambientes de produção.

A literatura explora diversas abordagens para integrar *Large Language Models*(LLMs) em atividades de *pentesting*, a exemplo dos projetos *PentestGPT* [Deng et al. 2024], *CIPHER* [Pratama et al. 2024] e *Pentest-AI* [Bianou and Batogna 2024], que evidenciam o potencial da linguagem natural na automação de testes de segurança. Diferentemente dessas propostas, focadas em exploração e intrusão, este trabalho propõe na etapa anterior que é a de reconhecimento e análise de vulnerabilidades.

A solução integra o OpenVAS [Gajula and Vassilakis 2024] a um agente baseado em linguagem natural, otimizando a configuração de varreduras e a análise de resultados. O objetivo central é simplificar a etapa de reconhecimento de vulnerabilidades, tornando-a mais prática e acessível. As contribuições deste trabalho são as seguintes:

- **Uma arquitetura de agente ReAct integrada ao GVM**, com três ferramentas especializadas (*TaskCreator*, *ResultAnalyzer* e *CSVAnalyzer*) orquestradas por um supervisor de decisão em *LangGraph*, descrita na Seção 3.
- **Uma camada híbrida de provedores de LLMs**, que permite alternar entre modelos proprietários e *open-source* sem alteração de código, endereçando restrições de custo e de soberania de dados (Seção 3.3).
- **Uma caracterização estruturada do esforço operacional** exigido pelas três interfaces disponíveis para a criação de uma varredura, explicitando que o ganho da interface conversacional não está na redução do número de interações — que permanece comparável — mas na eliminação do conhecimento prévio exigido sobre a taxonomia de entidades do GVM (Seção 4.3).
- **A disponibilização pública do código-fonte** da ferramenta sob licença MIT, permitindo o uso e a extensão da solução pela comunidade.

Cabe delimitar o escopo da avaliação: este trabalho não realiza estudo de usabilidade com usuários reais. A caracterização apresentada é estrutural, baseada nas propriedades das interfaces, e não sustenta afirmações quantitativas sobre tempo de execução, taxa de erro ou satisfação de usuários.

O agente é baseado na arquitetura *ReAct* [Yao et al. 2023] e utiliza o *framework LangChain* para conectar as *APIs* do *OpenVAS* a uma camada de inferência de LLMs flexível. Essa abordagem híbrida suporta modelos proprietários, como o *ChatGPT (OpenAI)*, e modelos *open-source* eficientes (e.g., Llama 3, Mixtral) executados via *Groq* para otimização de latência e custos. O sistema é composto de dois módulos principais: criação de varreduras via comandos naturais e análise automatizada de relatórios. Assim, o usuário configura e acompanha escaneamentos sem a necessidade de navegar pela interface técnica do *OpenVAS*, o que amplia a usabilidade da ferramenta para diferentes perfis.

A validação da ferramenta ocorreu por meio de um estudo comparativo focado em usabilidade e eficiência. A avaliação mensurou métricas quantitativas, como passos para execução de tarefas e complexidade de interação, confrontando o agente de Inteligência Artificial (IA) com as interfaces tradicionais: *Greenbone Security Assistant* (GSA) e *Greenbone Vulnerability Management - Command Line Interface* (GVM-CLI).

O artigo está organizado em cinco seções. A Seção 2 revisa os trabalhos relacionados no contexto da automação de varreduras de vulnerabilidades. A Seção 3 detalha a ferramenta proposta, incluindo arquitetura, tecnologias e demonstração de uso. A Seção 4 é dedicada à avaliação, descrevendo a metodologia e os resultados obtidos. Por fim, a Seção 5 apresenta as conclusões e sugere caminhos para trabalhos futuros.

2. Trabalhos Relacionados

Os recentes avanços dos *LLMs* possibilitam o uso de NLP para a criação de aplicações que implementam a integração de *LLMs* com *pentesting* (testes de penetração), visando a automação de *tasks* e uma melhor experiência de usuário com as ferramentas especializadas em análise de vulnerabilidades de redes (*Nmap*, *OpenVAS*, *Nessus*). Sendo assim, trabalhos recentes investigam essa abordagem, como *PENTESTGPT* [Deng et al. 2024], *CIPHER* [Pratama et al. 2024] e *PENTEST-AI* [Bianou and Batogna 2024], demonstrando o potencial de utilização de *LLMs* na execução de *pentesting*.

[Deng et al. 2024] apresentam o potencial dos LLMs na execução de testes de penetração, auxiliando na interpretação de *saídas* e proposição de ações subsequentes. Embora o trabalho tenha foco no *pentesting*, os autores demonstraram que *LLMs* podem ser úteis para analisar vulnerabilidades e sugerir priorizações baseadas em risco.

Da mesma forma, [Pratama et al. 2024] explora a utilização de um modelo refinado, onde foi feito um *fine tuning* e o *dataset* usado de forma superficial apenas para contexto no suporte a pesquisadores em segurança, estruturando a exploração de vulnerabilidades com base em *write-ups* documentados.

Considerando o ciclo de trabalho do *hacking* ético como descrito em [Yaacoub et al. 2021], composto pelas etapas de reconhecimento, varredura, ganho de acesso, manutenção de acesso e cobertura de rastros, optamos pelo uso do *OpenVAS*, concentrando o esforço nas primeiras duas etapas. Essa ferramenta oferece uma solução

abrangente e poderosa para varredura e gerenciamento de vulnerabilidades, sendo um dos *scanners* de segurança gratuitos mais robustos disponíveis, além de oferecer gerenciamento de falsos positivos nos resultados da varredura.

A solução proposta neste trabalho visa abordar as etapas de reconhecimento de vulnerabilidade e varredura de rede de forma autônoma, integrando *LLMs* com a ferramenta *OpenVAS*, de modo que o profissional de segurança possa, de forma mais eficiente e escalável, realizar a análise de sistemas. Enquanto soluções como PENTESTGPT [Deng et al. 2024] e CIPHER [Pratama et al. 2024] focam na automação do ciclo de *pentest* completo para especialistas, este trabalho aborda uma lacuna (*gap*) específica: a barreira de usabilidade de ferramentas de análise de vulnerabilidade.

A relevância de aplicar técnicas de IA e Processamento de Linguagem Natural (PLN) no domínio de segurança de redes é reforçada por trabalhos recentes onde autores têm explorado o uso de LLMs tanto para a construção quanto para a atualização de datasets de vulnerabilidades [Fideles et al. 2025]. Este contexto demonstra o alinhamento da nossa proposta, que foca na aplicação de LLMs para diminuir a barreira de usabilidade de ferramentas de análise, como o *OpenVAS*.

Por fim, este trabalho reforça a relevância de soluções baseadas em *LLMs* para a ampliação do acesso à análise de vulnerabilidades. As contribuições incluem não apenas a interface em linguagem natural e os ganhos de acessibilidade, mas também reflexões sobre possíveis aprimoramentos futuros, como o uso de agentes especializados e técnicas de *chunking* para lidar com relatórios extensos. Assim esta abordagem pode representar um novo patamar de usabilidade para ferramentas de segurança como o *OpenVAS*.

Tabela 1. Comparativo de Trabalhos Relacionados em IA e Segurança Ofensiva

Trabalho	Foco	Técnica IA	Escopo	Objetivo
PentestGPT	Pentesting e CTFs	Engenharia de Prompt	Geração de comandos e raciocínio	Exploração de falhas e intrusão
CIPHER	Assistente de Pentest	LLMs com fine-tuning	Sugestão de passos via *write-ups*	Guia para pentesters humanos
Pentest-AI	Pentest (MITRE)	Sistema multi-agente	Planejamento de etapas de ataque	Automação completa de intrusão
VulnSyncAI	Gestão de vulnerabilidades	PLN e LLMs modulares	Construção e atualização contínua de *datasets*	Curadoria de dados

A Tabela 1 sumariza as características das abordagens analisadas. Observa-se uma tendência clara na literatura em focar na automação de processos ofensivos complexos (*exploit* e intrusão), como visto no PentestGPT e Pentest-AI, ou na curadoria de dados brutos, como no VulnSyncAI.

Entretanto, nota-se a ausência de soluções focadas na usabilidade operacional de

ferramentas de varredura estabelecidas. Diferente dos trabalhos listados, que visam substituir o humano no ataque ou gerenciar bases de dados, a proposta deste trabalho se posiciona como um *middleware* inteligente. O foco é facilitar a interação com o *OpenVAS*, abstraindo a complexidade de configuração e interpretação de relatórios para facilitar o acesso à análise de vulnerabilidades, preenchendo a lacuna entre a ferramenta técnica e o operador não-especialista.

3. *OpenVAS Agent*

3.1. Visão Geral

O agente proposto foi desenvolvido com base na arquitetura *ReAct (Reasoning and Acting)* [Yao et al. 2023], que permite a um modelo de linguagem intercalar raciocínio e ações de forma iterativa. Em vez de apenas gerar respostas diretas, o agente ReAct analisa o contexto, toma decisões com base em raciocínio lógico e executa chamadas a ferramentas externas[Schick et al. 2023] — componentes funcionais integrados que realizam ações específicas fora do modelo, como acessar APIs, buscar informações ou manipular dados.

A proposta busca integrar três módulos explicados a seguir em uma solução baseada em IA, capaz de compreender instruções em linguagem natural, planejar a melhor sequência de ações e interagir com as ferramentas conforme necessário. Essa abordagem automatiza e simplifica a utilização do *OpenVAS*, promovendo acessibilidade mesmo para usuários sem conhecimento técnico avançado em segurança cibernética.

3.2. Arquitetura

O agente proposto foi desenvolvido com base na arquitetura ***Reasoning and Acting (ReAct)***, que permite ao modelo de linguagem intercalar raciocínio e execução de ações através de ferramentas externas. Esse ciclo iterativo ajuda a resolver *tasks* complexas que envolvem planejamento e interação com sistemas, como o *OpenVAS*.

Os componentes da Figura 1 apresentam o fluxo principal do agente:

- **1. Entrada do Usuário: (*Prompt*)**: O operador submete, em linguagem natural, a tarefa de segurança desejada (por exemplo, “Crie uma varredura na minha rede”), sem precisar de comandos técnicos.
- **2. Entrada (*Query*)**: A instrução é capturada pela interface do agente e encaminhada ao núcleo de processamento, disparando o ciclo de supervisão e raciocínio.
- **3. Interpretação e Raciocínio**: O agente utiliza LLMs para interpretar o objetivo da solicitação e definir um plano de ação interno.
- **4. Acesso a Ferramentas**: Conforme o plano, o agente invoca ferramentas integradas:
  - **(i) GVM API (*TaskCreator*)**: Cria e configura novas tarefas de varredura no *OpenVAS* a partir dos parâmetros do usuário.
  - **(ii) Provedores de LLMs (*ResultAnalyzer*)**: Recupera, interpreta e sintetiza resultados das varreduras, utilizando APIs como OpenAI ou Groq para gerar diagnósticos, resumos e recomendações.
  - **(iii) CSV do usuário (*CSVAnalyzer*)**: Ingere e analisa relatórios CSV com *Pandas DataFrame*, permitindo filtragens, estatísticas e identificação de padrões de vulnerabilidade em grandes volumes de dados.

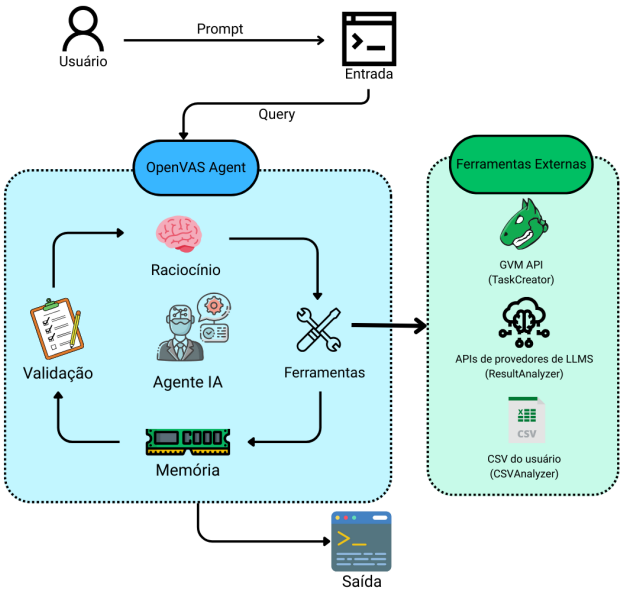

Figura 1. Arquitetura do *OpenVAS Agent*

- **5. Memória**: Armazena histórico de tarefas, resultados e decisões para otimizar interações futuras e manter consistência.
- **6. Validação**: Verifica se o resultado das ferramentas é coerente com a solicitação inicial; em caso de inconsistências, o raciocínio é reavaliado.
- **7. Saída**: A resposta final é estruturada em linguagem natural e apresentada ao usuário.

3.3. Tecnologias Utilizadas e Orquestração do Sistema

O sistema foi desenvolvido em Python, integrando as APIs do GVM (*Greenbone Vulnerability Management*) a provedores de inteligência artificial. A base do agente utiliza o framework *LangChain* [S et al. 2023] para construção de agentes do tipo *ReAct* [Yao et al. 2023], estendido pelo **LangGraph**, que organiza o fluxo em um grafo de estados cíclico, preservado no objeto `AgentState` como memória de curto prazo.

O grafo de execução é composto por nós especializados que acionam ferramentas do sistema:

- **Supervisor de Decisão**: Atua como "cérebro" do sistema, analisando o `AgentState` para decidir o próximo especialista ou encerrar a interação.
- **TaskCreator**: Cria e configura tarefas de varredura no *OpenVAS* [Aksu et al. 2019], extraindo parâmetros em linguagem natural e comunicando-se com o servidor via soquete Unix.
- **ResultAnalyzer**: Monitora o *status* das tarefas em execução no GVM e facilita o acompanhamento em tempo real.
- **CSVAnalyzer**: Processa relatórios exportados em CSV, reduzindo dados com *pandas* e gerando resumos para os LLMs [Muharrom and Saktiansyah 2023].

Para equilibrar desempenho, custo e soberania de dados, o módulo `main.py` implementa uma camada híbrida de provedores de LLMs. A solução permite alternar entre

modelos proprietários, como o **GPT (OpenAI)**, e modelos *open-source* de alta performance, como **Llama 3.3** e **Mixtral** via API da **Groq**, democratizando o uso do *OpenVAS* em diferentes contextos operacionais. O código está disponível no repositório do projeto[1]. Um exemplo de como rodar é `python3 main.py`. Fotos do funcionamento como, comandos de entrada, saídas da ferramentas estão disponíveis na seção Apêndice.

4. Avaliação

4.1. Objetivo da Avaliação

O objetivo principal deste estudo foi validar a eficácia do agente em reduzir a complexidade operacional do *OpenVAS*. A avaliação concentrou-se em medir o ganho de eficiência na execução de processos de segurança ofensiva, comparando a interação via linguagem natural com os métodos tradicionais (interface gráfica e linha de comando).

O foco da análise recaiu sobre dois eixos: a agilidade na configuração de varreduras (*tasks*) e a acessibilidade dos diagnósticos gerados. A medida buscada foi verificar se a abstração dos parâmetros técnicos do *OpenVAS* permite que usuários configurem escaneamentos e obtenham *insights* de segurança com menor número de interações e menor exigência de conhecimento prévio da ferramenta.

Dessa forma, a avaliação visa demonstrar a aplicabilidade prática da ferramenta em rotinas operacionais, quantificando a redução do esforço manual e da curva de aprendizado necessária para operar o framework de vulnerabilidades.

4.2. Critérios de Avaliação

Para avaliar o desempenho do agente frente às interfaces nativas do *OpenVAS*, adotamos critérios mistos focados na eficiência da interação e na qualidade dos resultados. A eficiência de interação mede a produtividade operacional contabilizando o esforço físico e cognitivo, como o número de cliques e comandos necessários para iniciar uma varredura. Já a praticidade operacional analisa a redução de barreiras para usuários não especialistas, comparando a navegação em menus complexos ou a memorização de sintaxes com a fluidez da linguagem natural. Por fim, a métrica qualitativa de precisão e clareza assegura que a simplificação do processo mantenha a integridade técnica, entregando diagnósticos corretos e de fácil compreensão.

4.3. Procedimento e Resultados

A avaliação quantitativa comparou o desempenho do agente com as interfaces nativas do *OpenVAS*: a interface gráfica (GSA) e a linha de comando (GVM-CLI). A Tabela 2 sintetiza as métricas de interação necessárias para a configuração de uma varredura padrão.

No GSA, o procedimento exige criar primeiro um *Target*, em tela distinta sob o menu *Configuration*, e em seguida uma *Task* sob o menu *Scans*, totalizando 11 interações (cliques) e o preenchimento manual de 5 campos. No GVM-CLI, o mesmo resultado requer 7 comandos encadeados, nos quais o identificador retornado por uma invocação é insumo da seguinte, exigindo domínio da sintaxe do protocolo GMP.

No agente, conforme a Figura 2, o usuário inicia com uma instrução de intenção (“quero criar uma *task*”) e o sistema conduz um diálogo guiado de cinco turnos adicionais,

[1] `https://github.com/phael-exe/CEIA_OpenVAS_Agent`

solicitando individualmente o nome da tarefa, a decisão de criar um novo alvo, o nome do alvo, o endereço do *host* e o índice da lista de portas — seis turnos no total.

Os dados evidenciam, portanto, que o número de interações é comparável entre as três interfaces, e que a contribuição da interface conversacional não reside na redução da contagem de passos. A diferença relevante está no conhecimento pressuposto: nas interfaces nativas, o usuário precisa saber de antemão que uma *Task* depende de um *Target*, que um *Target* depende de uma *Port List*, e onde cada uma dessas entidades é configurada. No agente, essa estrutura permanece implícita, pois cada parâmetro é solicitado no momento em que se torna necessário e a ordem de dependência é resolvida pelo sistema.

Cabe registrar uma contrapartida. A entrada por texto livre em assistentes conversacionais pode introduzir erros de digitação e carga cognitiva que a seleção em elementos gráficos estruturados não impõe. A magnitude desse *trade-off* não é mensurável pelo protocolo adotado neste trabalho e depende de avaliação com usuários reais.

Essa abstração é refletida na métrica de ”Necessidade de Conhecimento Técnico Prévio”. O agente interpreta a intenção do usuário e extrai automaticamente os parâmetros necessários (como alvo e tipo de scan) da frase enviada, reduzindo a barreira de entrada classificada como ”Alta”no CLI e ”Média”no GSA para um nível ”Baixo”.

Tabela 2. Comparativo do esforço operacional para criação de uma tarefa de varredura padrão. As unidades das três colunas não são diretamente comparáveis entre si; a tabela caracteriza a natureza do esforço exigido, não uma razão de desempenho.

Métrica de Avaliação	GSA (Web UI)	GVM-CLI	*OpenVAS* Agent (IA)
Interações discretas	11 cliques	7 comandos	6 turnos
Parâmetros informados	5 campos	4 parâmetros	5 respostas
Sintaxe a memorizar	Não	Sim (GMP)	Não
Conhecimento prévio da taxonomia do GVM	Necessário	Necessário	Dispensável
Parâmetros solicitados explicitamente	Não	Não	Sim

4.4. Análise Crítica

A solução proposta integra o *OpenVAS* a um agente de IA baseado em linguagem natural, estabelecendo uma camada de mediação entre a operação técnica e o usuário. Ao viabilizar a criação de *tasks* e a análise de relatórios via comandos textuais, o sistema altera a acessibilidade da análise de vulnerabilidades, estendendo-a a perfis com diferentes níveis de experiência, desde profissionais juniores até gestores.

A interface em linguagem natural atua na redução da complexidade operacional do *OpenVAS*. A implementação da arquitetura *ReAct*, em conjunto com a biblioteca *LangChain* e o modelo *GPT-4o mini*, possibilitou a automação de etapas de triagem, como a priorização de vulnerabilidades, a extração de recomendações e a identificação de falhas recorrentes.

No entanto, identificaram-se limitações técnicas inerentes aos LLMs. A restrição da janela de contexto impacta o processamento de relatórios extensos, comuns em auditorias de redes corporativas complexas. Nesses cenários, ocorre o truncamento de dados, o que pode excluir vulnerabilidades de menor severidade ou informações técnicas complementares da análise. Adicionalmente, quando a base de conhecimento não fornece evidências específicas, o modelo tende a gerar respostas genéricas, reduzindo o detalhamento técnico necessário para validações que exigem justificativas baseadas em evidência.

A arquitetura híbrida evidencia o *trade-off* entre modelos proprietários e *open-source*. Modelos proprietários como o GPT, demonstram capacidade de raciocínio lógico para contextos complexos, porém introduzem variáveis de custo e privacidade de dados. Em contrapartida, a integração de modelos *open-source* (como o Llama 3 via Groq) favorece a auditabilidade e a soberania dos dados, embora demande ajustes de engenharia para equiparar a precisão da inferência.

Em análise final, a proposta valida a aplicação de sistemas inteligentes assistivos na cibersegurança. A ferramenta não apenas automatiza processos, mas amplia a capacidade de interpretação e priorização frente a ameaças, impactando a eficiência operacional e os processos de gestão de vulnerabilidades.

5. Conclusão

Este estudo abordou a integração de agentes baseados em modelos de linguagem [Shen et al. 2024] ao *OpenVAS* como método para a automação da análise de vulnerabilidades. A implementação validou a viabilidade técnica de configurar *tasks* de varredura e interpretar relatórios via comandos em linguagem natural, alterando os requisitos de interação com a ferramenta. Os resultados indicam uma redução na barreira de entrada técnica, estendendo a capacidade de interpretação dos achados de segurança a perfis variados, desde profissionais juniores até sêniores.

Identificaram-se, contudo, restrições operacionais ligadas ao processamento de grandes volumes de dados, especificamente em *logs* extensos de escaneamentos complexos. A janela de contexto finita dos modelos de linguagem impõe limites à completude da análise nesses cenários.

Como trabalhos futuros, aponta-se a implementação de estratégias de segmentação automática (*chunking*) [Liu et al. 2023] para o processamento de relatórios volumosos, bem como o desenvolvimento de uma arquitetura de sistemas multiagentes para a cooperação em subtarefas especializadas. Adicionalmente, é sugerido o treinamento de modelos específicos (*fine-tuning*) em dados de cibersegurança para ampliar a robustez e a profundidade técnica das inferências.

Referências

Aksu, M. U., Altuncu, E., and Bicakci, K. (2019). A first look at the usability of openvas vulnerability scanner. In *Workshop on usable security (USEC)*.

Bianou, S. G. and Batogna, R. G. (2024). Pentest-ai, an llm-powered multi-agents framework for penetration testing automation leveraging mitre attack. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 763–770.

Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., and Rass, S. (2024). Pentestgpt: An llm-empowered automatic penetration testing tool.

Fideles, D. R., Lautert, D. P., Kreutz, D., and Quincozes, S. E. (2025). Vulnsyncai: Pln e llms para construção e atualização contínua de datasets de vulnerabilidades. In *Anais do XLIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. Sociedade Brasileira de Computação (SBC).

Gajula, M. R. and Vassilakis, V. G. (2024). Evaluating the performance open-source vulnerability scanners. In *2024 14th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 377–382.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2023). Lost in the middle: How language models use long contexts.

Muharrom, M. and Saktiansyah, A. (2023). Analysis of vulnerability assessment technique implementation on network using openvas. *International Journal of Engineering and Computer Science Applications (IJECSA)*, 2(2):53–62.

Odun-Ayo, I., Blessing, O., Martins, I., Owoka, E., Owoseni, T., and Omonedo, E. (2024). Comparative review of vulnerability analysis tools. In *2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG)*, pages 1–6.

Pratama, D., Suryanto, N., Adiputra, A. A., Le, T.-T.-H., Kadiptya, A. Y., Iqbal, M., and Kim, H. (2024). Cipher: Cybersecurity intelligent penetration-testing helper for ethical researcher. *Sensors*, 24(21):6878.

S, B. L., R, S. V., Mahadevan, R., and Raman, R. C. (2023). Comparative study and framework for automated summariser evaluation: Langchain and hybrid algorithms.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools.

Sharma, M., Desai, D., Arun, A. R., L, P., and Rajagopalan, N. (2024). Openvas vs the rest: Unveiling the competitive edge in vulnerability scanners. In *2024 3rd International Conference for Innovation in Technology (INOCON)*, pages 1–6.

Shen, X., Wang, L., Li, Z., Chen, Y., Zhao, W., Sun, D., Wang, J., and Ruan, W. (2024). Pentestagent: Incorporating llm agents to automated penetration testing.

Yaacoub, J.-P. A., Noura, H. N., Salman, O., and Chehab, A. (2021). A survey on ethical hacking: Issues and challenges.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). React: Synergizing reasoning and acting in language models.

Apêndice

Casos de Uso e Demonstração do Sistema

Para validar a eficácia da solução proposta na mediação da complexidade do *OpenVAS*, esta seção descreve três cenários de uso fundamentais que cobrem desde a configuração da varredura até a análise de dados brutos.

Automação de Varreduras com o TaskCreator

O fluxo operacional convencional para a criação de uma *task* no *Greenbone Security Assistant* (GSA) exige que o usuário navegue por múltiplos menus para configurar alvos (*Targets*) e selecionar o tipo de escaneamento (*Scan Config*). Como demonstrado na Figura 2, o *OpenVAS Agent* simplifica este processo por meio de uma interface de chat.

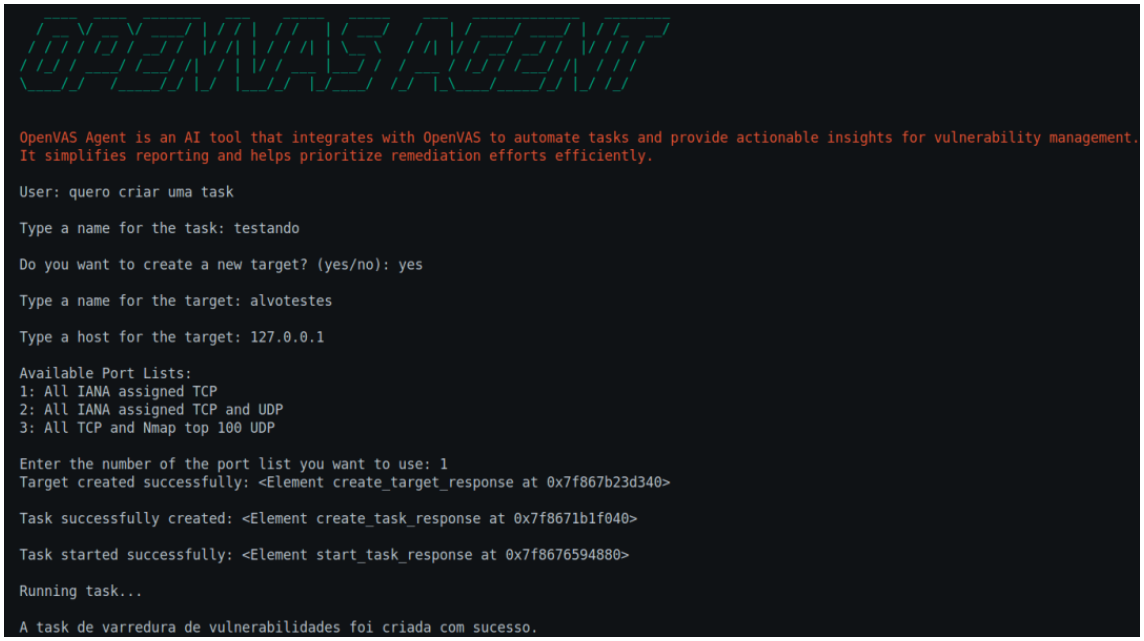

Figura 2. Imagem de funcionamento do *OpenVAS Agent*

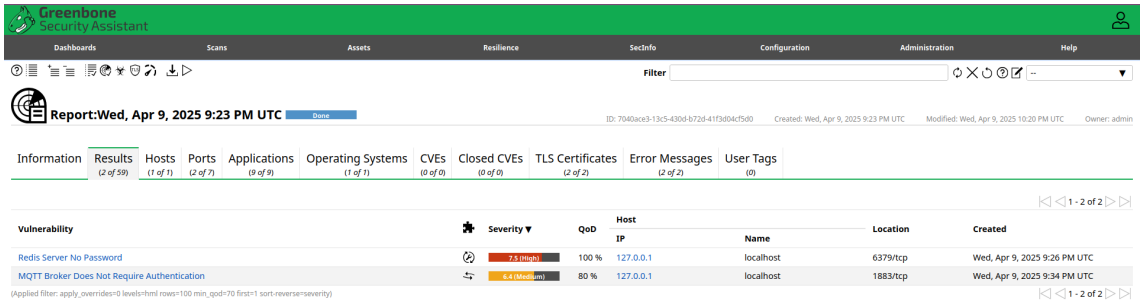

Figura 3. Imagem do resultado da varredura na interface visual do GSA pós criação de task via TaskCreator

Ao receber uma instrução como ”Escaneie o IP 192.168.1.10 em busca de vulnerabilidades”ou ”Quero criar uma *task*”, o agente utiliza o framework LangChain para orquestrar a criação do alvo e a execução da tarefa via protocolos de API. A Figura 3 ilustra o resultado dessa ação refletido diretamente na interface visual do GSA. A Figura 2 explicita o comportamento passo a passo do assistente: a instrução inicial de intenção é seguida da coleta individual do nome da tarefa, da confirmação de criação de novo alvo, do nome do alvo, do endereço do *host* e do índice da lista de portas, totalizando seis turnos de diálogo.

Interpretação de Resultados via ResultAnalyzer

Após a conclusão de uma varredura, o volume de dados técnicos pode ser proibitivo para usuários com menor expertise. O componente *ResultAnalyzer* atua na camada de pós-processamento, consultando os resultados da *task* gerada anteriormente conforme mostra a Figura 4.

Figura 4. Imagem do *ResultAnalyzer* sobre a task gerada pelo *TaskCreator*

O agente não apenas lista as falhas, mas as contextualiza utilizando o raciocínio *ReAct*. Ele identifica criticidades e sugere prioridades de mitigação em linguagem natural. Em vez de analisar manualmente uma tabela de CVEs, o analista recebe um diagnóstico direto sobre o estado de segurança do ativo, permitindo uma tomada de decisão mais ágil e fundamentada.

Análise de Dados Estruturados com o CSV Analyzer

Complementando as funções de varredura direta, a solução oferece o *CSV Analyzer*, uma funcionalidade voltada para a análise de datasets de vulnerabilidades ou relatórios exportados de fontes externas. A Figura 5 demonstra o funcionamento desta ferramenta.

Diferente da análise de relatórios estáticos, o *CSV Analyzer* permite que o usuário faça perguntas complexas sobre o conjunto de dados, como: “Qual é a média de score CVSS para vulnerabilidades de rede encontradas no arquivo?”. O agente processa o arquivo CSV, realiza as agregações necessárias e apresenta a resposta de forma interpretativa. Esta funcionalidade alinha-se à proposta do *VulnSyncAI* ao utilizar LLMs para extrair conhecimento útil de grandes volumes de dados de cibersegurança, eliminando a necessidade de ferramentas de manipulação de dados externas ou conhecimento em linguagens de consulta.

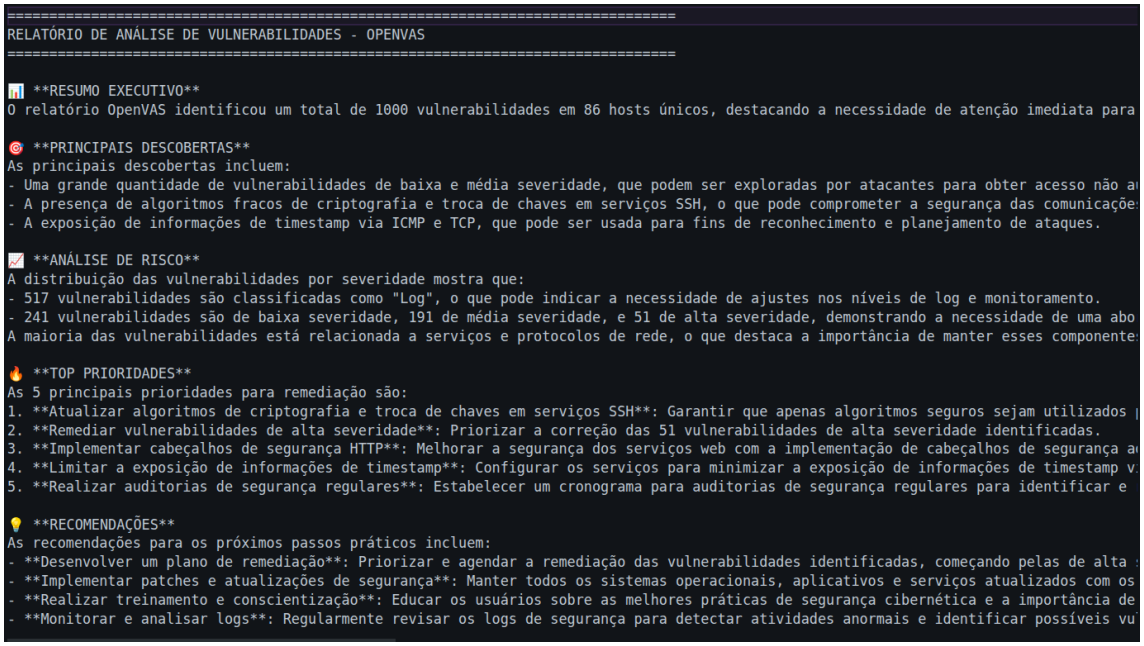

Figura 5. Imagem do *CSVAnalyzer* sobre um arquivo de testes de *tasks* em lote