PRÉMIOS DE SEGURANÇA
SECURITY MAGAZINE | REVISTA DOS PROFISSIONAIS DE SEGURANÇA

SEPTA: Self-Evolving PenTester Agent

Rui Fernandes

SEPTA (Self-Evolving PenTester Agent) é um agente modular e autônomo para Hacking Ético e Testes de Penetração, construído sobre um LLM de código aberto (Llama 3.2) e uma estrutura de raciocínio no estilo ReAct. O sistema compreende três módulos integrados: Módulo LLM PenTester — A camada central de interação entre o usuário e a base de conhecimento. O agente recebe consultas em linguagem natural e as classifica em três categorias: ataque (realização de varreduras de vulnerabilidades ou exploração de um alvo), teach (orientar o usuário por etapas de hacking ético com criação opcional de sandbox) e informação (responder consultas usando documentação recuperada ou busca na web). O agente utiliza um banco de dados vetorial FAISS para armazenar e recuperar conhecimento relevante, incluindo documentação PDF, histórico de chat e resumos obtidos na web. Módulo Autoevolutivo — Em vez de reeducar ou ajustar o LLM base (o que é computacionalmente caro e corre o risco de degradar o modelo), o SEPTA atualiza continuamente seu conhecimento por meio do banco de dados vetorial. O agente ReAct pode buscar autonomamente na web (via API DuckDuckGo) por CVEs recentes e técnicas de ataque, avaliar sua relevância e armazenar conteúdo validado no banco de dados vetorial. Os usuários também podem injetar manualmente documentos ou textos para enriquecer a base de conhecimento. Isso mantém o sistema atualizado com os cenários de ameaças em evolução, sem exigir retreinamento de modelos. Módulo de Emulação — O SEPTA pode gerar e implantar ambientes de rede virtual de forma autônoma usando Vagrant e VirtualBox. As ações disponíveis incluem: GenerateVagrantfile (produz uma configuração completa de VM para alvos no Windows, Linux ou macOS), CreateAndRunVagrant (instancia e inicia o sandbox) e RunCmdInVM (executa comandos como varreduras nmap ou exploits Metasploit dentro da VM em execução). Isso permite que o agente valide seu próprio conhecimento executando comandos em um ambiente seguro e isolado e observando resultados reais. Ao final de uma sessão de ataque, o agente pode gerar um relatório estruturado de vulnerabilidade via GenerateVulnerabilityReport, detalhando as vulnerabilidades identificadas com avaliações de impacto de confidencialidade, integridade e disponibilidade. Juntos, esses módulos permitem que o SEPTA sirva como uma plataforma de treinamento para profissionais de cibersegurança, um assistente autônomo de pentesting e um sistema de conhecimento autocorretivo — tudo em uma solução integrada e única.

O cenário da cibersegurança está evoluindo em um ritmo que supera a capacidade humana de responder. O número de Vulnerabilidades e Exposições Comuns (CVEs) recém-divulgadas supera consistentemente a disponibilidade de patches de segurança, e relatórios recentes da ENISA indicam que o hacktivismo e as ameaças cibernéticas à infraestrutura crítica estão crescendo em volume e imprevisibilidade. Grandes Modelos de Linguagem surgiram como ferramentas poderosas para tarefas de cibersegurança, incluindo testes de penetração. No entanto, sua adoção em ambientes de produção é dificultada por duas limitações críticas: alucinações (resultados confiantes, porém incorretos) e estagnação do conhecimento (incapacidade de raciocinar sobre vulnerabilidades recém-descobertas). Agentes de pentesting baseados em LLMs existentes requerem intervenção humana frequente para corrigir erros, reduzindo os ganhos de produtividade que deveriam proporcionar. Além disso, a maioria dos sistemas atuais é comparada com conjuntos de dados estáticos, que não capturam variabilidade real em configurações de rede e cenários de ataque. Há uma falta de ambientes de emulação realistas e dinâmicos onde um agente possa testar, validar e atualizar continuamente seu próprio conhecimento. O SEPTA foi desenvolvido para preencher essas

lacunas: criar um agente de pentesting que não apenas auxilie os usuários em tarefas éticas de hacking, mas também seja capaz de reconhecer os limites do próprio conhecimento, buscar informações atualizadas de forma autônoma, validá-las em um ambiente real emulado e incorporá-las à sua base de conhecimento — sem exigir retreinamento de modelos. Essa abordagem mira tanto o problema da precisão (alucinações) quanto o problema da moeda (conhecimento desatualizado da ameaça), tornando o agente progressivamente mais confiável ao longo do tempo.

Um protótipo funcional do SEPTA foi validado em um servidor equipado com um processador Intel Xeon E5-2690 v2 @ 3,00 GHz e 120 GB de RAM, rodando Python 3.10 e Llama 3.2. A base de conhecimento do RAG foi semeada com três documentos técnicos (documentação do Vagrant, VirtualBox e Metasploit). Em um cenário de teste representativo — onde o usuário solicitou um laboratório de prática de hacking ético contra uma máquina Windows com vulnerabilidade simulada — o agente demonstrou a seguinte sequência de decisão autônoma: Invocou o CreateAndRunVagrant para instanciar duas VMs (atacante Kali + alvo do Windows) na mesma rede virtual. Reconheceu uma lacuna em seu conhecimento sobre vulnerabilidades Metasploitable2 e acionou autonomamente o WebSearchAbstract. Após a confirmação da relevância pelo usuário, armazenava o resumo recuperado via StoreAbstract. Invocado o RunCmdInVM para executar uma varredura de porta nmap contra o IP alvo, identificando corretamente uma porta HTTP aberta. Produziu uma resposta final coerente guiando o usuário para a próxima etapa de exploração. Qualitativamente, a sequência de tomada de decisão do agente foi avaliada como consistente com o comportamento de pentesting de especialistas, sequenciamento correto do reconhecimento antes da exploração e autocorreção de lacunas de conhecimento antes de agir. O loop ReAct operava em um máximo de sete iterações por consulta. A principal limitação identificada do sistema é o tempo de configuração da emulação para configurações complexas de rede, que é uma infraestrutura e não uma restrição algorítmica.

Vários agentes de pentesting baseados em LLM foram propostos na literatura acadêmica, incluindo VulnBot, ARACNE, RapidPen, AutoAttacker e PenTest++. Embora esses sistemas demonstrem fortes resultados na geração de comandos e na exploração autônoma, compartilham uma limitação estrutural comum: seu conhecimento é estático no momento da inferência. Uma vez implantados, eles não conseguem identificar e corrigir autonomamente lacunas em seu próprio conhecimento sem atualizações impulsionadas por humanos ou retreinamento completo do modelo. O SEPTA se diferencia em três dimensões: Arquitetura do conhecimento autoevolutiva — Ao contrário de sistemas concorrentes que dependem de bancos de dados RAG fixos ou ajuste fino estático, o agente do SEPTA detecta de forma autônoma déficits de conhecimento, recupera informações atualizadas da web, valida por emulação e persiste no banco de dados vetorial. Isso cria um ciclo de retroalimentação que melhora progressivamente a confiabilidade do agente ao longo do tempo, sem custos de requalificação. Ambiente integrado de emulação — A maioria dos agentes existentes opera contra alvos vulneráveis pré-configurados (por exemplo, Hack The Box, Metasploitable). O SEPTA pode gerar e provisionar autonomamente o próprio ambiente-alvo usando o Vagrant, permitindo a criação flexível e sob demanda de cenários. Isso é particularmente relevante para aplicações de treinamento onde reproduzibilidade e diversidade de cenários são necessárias. Código aberto e acessível a preços acessíveis — o SEPTA é construído inteiramente sobre componentes de código aberto (Llama 3.2, LangChain, FAISS, Vagrant, VirtualBox), tornando-o implantável sem dependência de serviços pagos de API como GPT-4 ou Gemini, que são usados pela maioria dos sistemas concorrentes. Isso é um diferencial significativo para instituições acadêmicas, PMEs e equipes de segurança do setor público que operam sob restrições orçamentárias.

O SEPTA integra múltiplas camadas avançadas de tecnologia: Large Language Model (LLM) — Llama 3.2 (Meta), um modelo de fundação open-source, serve como motor de raciocínio. Ele atua localmente, preservando a confidencialidade dos dados. Framework do Agente ReAct — Implementado via LangChain, o paradigma de prompting ReAct (Raciocínio + Atuação) permite que o agente decomponha entradas complexas do usuário em cadeias de pensamento, selecione ferramentas apropriadas, observe resultados e revise seu plano iterativamente — reduzindo significativamente alucinações em comparação com a inferência de LLM de passagem única. Geração Aumentada por Recuperação (RAG) — O banco de dados vetorial FAISS (Facebook AI Similarity Search) oferece recuperação semântica de alta velocidade do conhecimento armazenado, permitindo que o agente fundamente suas respostas em documentação verificada, em vez de depender apenas do conhecimento do modelo paramétrico. Virtualização Automatizada — Vagrant e VirtualBox são usados para gerar, provisionar e gerenciar programaticamente ambientes de rede virtual multi-máquina, possibilitando execução segura de comandos e emulação de cenários. Integração com Busca na Web — A integração da API DuckDuckGo permite que o agente recupere autonomamente informações atuais de inteligência de ameaças e CVE, alimentando o ciclo de autoevolução. Python 3.10 foi usado para a pilha completa de implementação, garantindo ampla compatibilidade e extensibilidade.

O projecto contou com a Bolsa de investigação para aluno de doutoramento. - Financiado com financiamento nacional pela FCT por meio da bolsa individual de pesquisa PRT/BD/154919/2023 com o correspondente DOI https://doi.org/10.54499/PRT/BD/154919/2023

O projecto foi desenvolvido em colaboração com o Instituto Politécnico do Cávado e do Ave, 2AI Research Center

O protótipo foi desenvolvido e testado na Escola de Tecnologia do IPCA (2Ai), em Barcelos, Portugal. Os testes foram realizados em infraestrutura dedicada de servidores de pesquisa (Intel Xeon E5-2690 v2, 120 GB de RAM). Marcos importantes incluíram: (1) integração do agente ReAct com emulação RAG baseada em FAISS e Vagrant; (2) validação do ciclo autônomo de tomada de decisão entre cenários de ataque, ensino e informação; (3) demonstração de provisionamento autônomo de VMs de ponta a ponta, descoberta de vulnerabilidades e exploração guiada. O desenvolvimento e os testes ocorreram dentro do escopo do ciclo de pesquisa acadêmica 2024–2025. Após os insights obtidos durante a fase piloto de testes, uma nova e aprimorada versão do SEPTA está atualmente em desenvolvimento ativo e deve ser finalizada no primeiro semestre de 2026.