ARPIA – FRAMEWORK INTERATIVO PARA INTEGRAÇÃO DE FERRAMENTAS DO KALI LINUX: UMA ABORDAGEM ACESSÍVEL PARA USUÁRIOS SEM EXPERIÊNCIA EM CIBERSEGURANÇA

1º SGT – ROBSON ARCANJO MENESES
1º SGT – FELIPE CÉSAR PEDROSA DE SOUSA
1º SGT – ROMERITO DORNELES ROCHA
1º SGT – CLEYTON FERNANDO DE ALMEIDA SILVA
1º SGT – JEAN SILVA DE LIMA

RESUMO

Este trabalho apresenta o desenvolvimento do Arpia, um framework em Python/Django que integra os principais módulos do Kali Linux voltados à segurança cibernética, com o objetivo de simplificar o uso dessas ferramentas e proporcionar uma experiência unificada e acessível. Tradicionalmente, as ferramentas do Kali Linux são operadas via linha de comando e demandam conhecimento técnico avançado, o que limita sua adoção por iniciantes. O Arpia busca superar essa barreira por meio de uma interface web interativa, que centraliza funções de varredura, análise de vulnerabilidades, pentest, coleta de evidências, integração com inteligência artificial e geração de relatórios automatizados. A pesquisa foi conduzida em ambiente controlado de rede virtualizada, utilizando o Proxmox e a máquina vulnerável Metasploitable 2, permitindo validar a integração e desempenho dos módulos implementados. Os resultados indicam que o Arpia mantém a precisão técnica das ferramentas originais, ao mesmo tempo em que reduz a complexidade operacional e melhora a usabilidade. Embora ainda demande otimizações de desempenho e testes em ambientes reais, o framework mostra-se promissor para aplicações educacionais, laboratoriais e operacionais em segurança da informação.

Palavras-chave: Cibersegurança, Framework, Integração.

1 INTRODUÇÃO

A segurança cibernética consolidou-se como um dos maiores desafios da era digital, exigindo soluções eficazes para identificar vulnerabilidades, realizar testes de intrusão e coletar evidências digitais. Nesse contexto, o Kali Linux destaca-se como uma das distribuições mais completas voltadas à segurança ofensiva, oferecendo um conjunto extenso de ferramentas para auditoria e análise forense. Contudo, seu uso demanda conhecimento técnico avançado e familiaridade com a linha de comando, o que limita a adoção por usuários iniciantes e dificulta o aprendizado prático.

Diante desse cenário, este trabalho propõe o desenvolvimento do Arpia, uma aplicação web baseada no framework Django, em Python, cujo objetivo é integrar, em uma única interface visual, as principais ferramentas do Kali Linux. A proposta visa simplificar o acesso e o uso dessas ferramentas, tornando-as mais acessíveis a profissionais e estudantes, sem comprometer sua robustez e confiabilidade. Além de contribuir para o aprendizado em segurança cibernética, o Arpia pretende facilitar a execução de testes e análises em ambientes controlados.

O problema central que norteia este estudo é: como integrar, de forma prática e funcional, as ferramentas do Kali Linux em um ambiente web, mantendo sua eficiência e segurança operacional.

O objetivo geral é desenvolver um framework integrador que permita a execução dessas ferramentas via interface gráfica, otimizando a interação entre usuário e sistema.

A estrutura deste trabalho está organizada nas seguintes partes: Resumo, Introdução, Desenvolvimento (composto por Revisão da Literatura, Métodos de Pesquisa, Apresentação e Análise de Dados e Discussão dos Resultados), Conclusão, Abstract e Referências. Ao final, são apresentados os resultados obtidos nos testes práticos realizados, demonstrando o potencial do Arpia como ferramenta de apoio à segurança da informação.

1.1 CONTEXTUALIZAÇÃO DO ESTUDO

A crescente digitalização de processos, serviços e relações sociais tem ampliado a exposição de sistemas computacionais a ameaças cibernéticas. Nesse cenário, a segurança da informação tornou-se estratégica para empresas, governos e usuários individuais. Ataques como vazamento de dados, ransomware e invasões a redes privadas tornaram-se frequentes e sofisticados, exigindo profissionais qualificados e ferramentas eficazes.

O Kali Linux consolidou-se como referência em auditoria e testes de penetração, por reunir ferramentas para análise de redes, exploração de vulnerabilidades e coleta de evidências digitais. Entretanto, sua operação exige domínio técnico e comandos específicos, o que dificulta sua utilização por iniciantes. Assim, o desenvolvimento de interfaces gráficas centralizadas representa um avanço, ao tornar tais recursos mais acessíveis e aplicáveis em contextos de aprendizado e operação prática.

1.2 JUSTIFICATIVA

A escolha por desenvolver um sistema web que centralize as ferramentas do Kali Linux se justifica pela lacuna existente entre a complexidade dessas ferramentas e a necessidade crescente de ampliar o acesso ao conhecimento prático em segurança cibernética. Muitos estudantes, entusiastas ou profissionais iniciantes enfrentam barreiras técnicas ao tentar utilizar essas ferramentas de forma eficiente, o que pode atrasar seu processo de formação ou limitar sua atuação.

Além disso, a fragmentação da interface do Kali Linux contribui para uma experiência operacional pouco intuitiva, o que pode prejudicar a produtividade mesmo de usuários mais experientes. Um sistema que reúna as principais funcionalidades em um ambiente visual, amigável e de fácil navegação, promovendo o aprendizado prático de forma mais acessível.

A utilização do Django como base para o desenvolvimento do framework oferece uma estrutura segura, escalável e bem documentada, o que contribui para a estabilidade e manutenção do sistema.

1.3 DEFINIÇÃO DO PROBLEMA DE PESQUISA

Integrar, em uma plataforma web unificada, as principais ferramentas do Kali Linux voltadas à segurança cibernética, de forma prática e eficiente, mantendo a robustez, funcionalidade e acessibilidade das operações, especialmente para usuários com pouca experiência técnica.

1.4 OBJETIVOS DA PESQUISA

1.4.1 OBJETIVO GERAL

Desenvolver e implantar framework simplificado que integre a varredura de vulnerabilidades em um ambiente laboratorial, com interface gráfica web para orquestração e visualização de resultados, visando identificar, validar e priorizar riscos cibernéticos no SC²EX.

1.4.2 OBJETIVOS ESPECÍFICOS

- identificar e selecionar as ferramentas mais relevantes do Kali Linux para integraçã no sistema;
- projetar e implementar a arquitetura backend do framework utilizando Django;

- desenvolver uma interface gráfica que facilite a interação com os recursos disponíveis;
- realizar testes práticos em ambiente controlado para validar a funcionalidade do sistema;
- avaliar a usabilidade e o desempenho do framework a partir de cenário simulado.

1.5 ESTRUTURA DO CONTEÚDO ESCRITO

A estrutura deste trabalho compreende: Resumo, Abstract, Referências e três partes principais, apresentadas a seguir.

1. Introdução: Contém a contextualização do tema, a justificativa da pesquisa, a definição do problema, os objetivos e a estrutura geral do documento.

2. Desenvolvimento, divide-se em quatro partes principais:

2.1 Revisão da Literatura: Aborda os conceitos teóricos fundamentais, como segurança cibernética, testes de penetração, uso do Kali Linux e desenvolvimento com Django.

2.2 Metodologia: Descreve os métodos e etapas utilizados no desenvolvimento do framework, incluindo ferramentas, ambiente de testes e critérios de validação.

2.3 Apresentação e Análise de Dados: Exibe os resultados obtidos durante a implementação e testes do sistema, com análises qualitativas.

2.4 Discussão dos Resultados: Avalia os impactos, limitações, possíveis melhorias do framework proposto e sugestões para trabalhos futuros.

3. Conclusão: Retoma os objetivos, apresenta os principais resultados alcançados, limitações enfrentadas.

2 DESENVOLVIMENTO

2.1 REVISÃO DA LITERATURA

2.1.1 SEGURANÇA CIBERNÉTICA

De acordo com Andress (2020), a segurança cibernética envolve práticas, tecnologias e processos voltados à proteção de sistemas, redes e dados contra acessos e ataques não autorizados. Com o avanço da transformação digital, tornou-se uma área estratégica para empresas e governos, dado o aumento exponencial no número de ameaças digitais. Ataques como ransomware, phishing e exploração de vulnerabilidades têm exigido respostas cada vez mais rápidas e eficazes (ANDRESS, 2020).

Segundo a norma ISO/IEC 27001 (2013), a segurança da informação deve garantir os pilares da confidencialidade, integridade e disponibilidade dos dados. Nesse contexto, as atividades de testes de penetração (pentest) e auditorias de segurança são fundamentais, pois permitem identificar falhas exploráveis antes que sejam utilizadas por agentes maliciosos.

Com o crescimento das infraestruturas críticas conectadas à internet, a cibersegurança passou a ser tratada também como um componente essencial da resiliência organizacional. Relatórios recentes apontam que o número médio de violações de dados aumentou mais de 25% entre 2022 e 2024, com custos médios superiores a 4,45 milhões de dólares por incidente (IBM, 2024). Essa realidade demonstra a necessidade de investimentos contínuos em prevenção, resposta e mitigação de incidentes.

Além dos princípios clássicos da segurança, novas abordagens vêm sendo incorporadas ao campo, como o conceito de Zero Trust, que pressupõe a ausência de confiança implícita em qualquer usuário ou sistema, exigindo autenticação contínua e m nitoramento constante (NIST, 2023). O m delo de defesa em profundidade e a análise comportamental de tráfego sã exemplos de estratégias aplicadas para reduzir a superfície de ataque.

A segurança cibernética contemporânea não se limita à proteção técnica, mas envolve também aspectos humanos, legais e organizacionais. Políticas de conscientização, governança e cultura de segurança tornam-se tão importantes quanto

as soluções tecnológicas empregadas. Como ressalta Grimes (2022), “a segurança é tão forte quanto o elo mais fraco e, na maioria dos casos, esse elo é o fator humano”.

2.1.2 KALI LINUX

O Kali Linux é uma distribuição baseada em Debian, desenvolvida com foco em segurança da informação, sendo amplamente utilizada para testes de penetração, análise forense e engenharia reversa. Possui mais de 600 ferramentas voltadas à segurança ofensiva e defensiva, entre elas Nmap, Wireshark e Burp Suite (COOPER, 2019).

De acordo com a Offensive Security (2024), responsável pelo seu desenvolvimento e manutenção, o Kali Linux é uma evolução direta do antigo BackTrack, lançado em 2006, e tornou-se o principal sistema operacional voltado para pentesting e avaliação de vulnerabilidades. Ele fornece suporte nativo para diversas arquiteturas e ambientes, como ARM, contêineres e WSL (Windows Subsystem for Linux), o que amplia sua aplicabilidade em diferentes cenários de teste.

A estrutura modular do Kali permite que cada ferramenta seja instalada e atualizada de forma independente, garantindo flexibilidade e segurança no ambiente de testes. Além disso, o sistema é projetado para operar em modo “live”, sem necessidade de instalação permanente, o que é útil para análises forenses e testes em campo (OFFENSIVE SECURITY, 2024).

Apesar de sua robustez, o Kali Linux apresenta limitações no que se refere à usabilidade, especialmente para usuários sem experiência avançada. A maioria das ferramentas exige familiaridade com a linha de comando e com os conceitos técnicos de segurança, o que pode dificultar sua aplicação em contextos educacionais e operacionais menos especializados (KALI LINUX, 2023).

Como observa Junaid (2025), utilitários como o Searchsploit demonstram a complexidade inerente do Kali: “Searchsploit é um utilitário de linha de comando no Kali Linux que permite aos usuários procurar explorações e vulnerabilidades no Banco de Dados de Exploração (Exploit-DB). Ele fornece uma cópia local do Exploit-DB, permitindo pesquisas off-line para explorações, shellcodes e papéis relacionados a vulnerabilidades de segurança” (JUNAID, 2025). Essa característica reflete a ênfase da distribuição em eficiência e controle total sobre o ambiente, em detrimento da facilidade de uso para iniciantes.

2.1.3 DESENVOLVIMENTO COM O DJANGO

O Django é um framework web de alto nível baseado em Python, que permite o desenvolvimento rápido e seguro de aplicações web. Ele segue o padrão Model-View-Template (MVT) e fornece diversas funcionalidades integradas, como autenticação de usuários, administração automatizada e proteção contra ataques comuns (DJANGO SOFTWARE FOUNDATION, 2023).

Segundo Tang e Kim (2023), o Django adota uma filosofia de desenvolvimento que prioriza a reutilização de código, a segurança e a escalabilidade. Seu sistema de ORM (Object-Relational Mapping) simplifica o acesso ao banco de dados e reduz o risco de injeções SQL, enquanto o uso de tokens CSRF e middleware de segurança integrados protege contra ataques baseados em sessão.

Comparado a outros frameworks Python, como Flask e FastAPI, o Django oferece um ecossistema mais completo e maduro, o que o torna particularmente adequado para aplicações corporativas e acadêmicas que exigem robustez e conformidade com boas práticas de segurança.

N contexto da segurança cibernética, Django oferece uma estrutura robusta para o desenvolvimento de plataformas que integram diferentes ferramentas, permitindo o gerenciamento de processos em interfaces amigáveis. Sua arquitetura modular e escalável o torna uma opção viável para projetos que envolvem a manipulação de

dados sensíveis e a execução de comandos complexos via backend.

Além disso, o framework incorpora recomendações de segurança baseadas em guias como o OWASP Top 10, o que reforça sua adequação a sistemas voltados à cibersegurança (OWASP, 2024). Essa combinação de segurança, flexibilidade e produtividade justifica sua escolha como base para o desenvolvimento do framework Arpia.

2.1.4 INTEGRAÇÃO DE SISTEMAS COM PYTHON

A linguagem Python tem se destacado como uma das mais utilizadas no desenvolvimento de aplicações voltadas à segurança da informação. Sua sintaxe simples, legibilidade e vasta gama de bibliotecas tornam-na ideal para automação de tarefas, desenvolvimento de scripts de análise e integração entre sistemas (ZANDER; JUDGE; SHARP, 2021).

Com o suporte a bibliotecas como subprocess, paramiko, os, requests, e scapy, Python possibilita a execução de comandos de sistemas operacionais, controle de processos externos e comunicação com protocolos de rede, como SSH, HTTP e ICMP. Essas características permitem que a linguagem funcione como um middleware entre sistemas distintos, possibilitando automação e integração de alto nível.

Em ambientes de segurança, Python é amplamente utilizado para desenvolver ferramentas de coleta de logs, análise de pacotes e exploração de vulnerabilidades. Frameworks como Impacket e Pwntools são exemplos de como a comunidade de segurança explora seu potencial para tarefas avançadas (MITCHELL; ZHANG, 2022).

Essas funcionalidades permitem que Python atue como uma "ponte" entre diferentes ferramentas de segurança, tornando viável a criação de frameworks que orquestram testes de penetração, varreduras e análises forenses a partir de uma interface centralizada. Essa capacidade de integração é especialmente útil na construção de sistemas que visam simplificar o uso de ferramentas complexas, como as disponíveis no Kali Linux (BEAZLEY, 2015).

Além da integração técnica, Python oferece um ecossistema maduro para desenvolvimento de aplicações seguras, com suporte a bibliotecas de criptografia, autenticação e controle de acesso, como PyCrypto, Cryptography e OAuthLib. Esse conjunto de recursos torna a linguagem uma escolha estratégica para a implementação de frameworks como o Arpia, voltados à automatização e acessibilidade das práticas de cibersegurança.

2.2 MÉTODOS DE PESQUISA

A metodologia adotada nesta pesquisa é de natureza aplicada, com abordagem qualitativa e caráter exploratório, pois busca propor e avaliar uma solução tecnológica prática, voltada à integração de ferramentas de segurança cibernética em um único ambiente funcional. Segundo Gil (2017), a pesquisa aplicada tem como finalidade gerar conhecimento para uso imediato na solução de problemas específicos, enquanto a abordagem qualitativa permite compreender fenômenos sob a perspectiva dos sujeitos e dos contextos em que se inserem.

Foram utilizados como procedimentos técnicos: a pesquisa científica, experimental e o estudo de caso. A pesquisa científica fundamentou-se em revisão bibliográfica e documental sobre as principais tecnologias envolvidas — Python, Django e Kali Linux —, de modo a embasar conceitualmente as decisões de projeto e integração. Já o caráter experimental consistiu na implementação prática de um framework funcional, denominado Arpia, validado por meio de experimentos em ambiente controlado.

O objetivo central consiste no desenvolvimento de um framework web integrador que permita a execução de ferramentas do Kali Linux por meio de uma interface gráfica acessível, utilizando o framework Django como base para a implementação. Essa proposta se justifica pela

necessidade de reduzir a complexidade operacional inerente ao uso das ferramentas do Kali Linux, normalmente dependentes da linha de comando, o que representa uma barreira para usuários sem ampla experiência técnica.

De acordo com Prodanov e Freitas (2013), a pesquisa exploratória é adequada quando o tema é pouco estudado ou quando se busca compreender um problema sob novas perspectivas. Nesse sentido, a criação do Arpia representa uma abordagem inovadora, que une a segurança ofensiva tradicional a conceitos de usabilidade e integração modular — temas raramente combinados em frameworks acadêmicos de cibersegurança.

Inicialmente, realizou-se uma pesquisa exploratória com o propósito de identificar as ferramentas mais relevantes voltadas à realização de análise de vulnerabilidades e coleta de evidências digitais. Essa etapa fundamentou-se tanto na revisão de literatura técnica quanto na experiência empírica dos autores, possibilitando a seleção das ferramentas mais adequadas aos objetivos do projeto. A escolha privilegiou utilitários amplamente reconhecidos no ecossistema do Kali Linux, como Nmap, Searchsploit e Rustscan, considerando fatores com precisão, eficiência e compatibilidade com ambiente Python.

O processo de desenvolvimento da solução foi estruturado em três etapas principais, detalhadas no Apêndice A – Instalação, Configuração e Operação do Framework Arpia, que descreve as ferramentas, o ambiente de testes e os critérios de validação:

a) Implementação do backend: as ferramentas selecionadas foram integradas ao backend da aplicação por meio de scripts desenvolvidos em Python, utilizando comandos nativos do Kali Linux e scripts de otimização dos resultados. A integração seguiu princípios de modularidade e isolamento, garantindo que falhas em um módulo não comprometessem funcionamento do sistema como um todo.

b) Desenvolvimento da interface web: uma interface gráfica foi projetada com foco em usabilidade, clareza e acessibilidade. Para sua construção, utilizaram-se tecnologias como HTML5, CSS3, JavaScript, o framework Django e o Python, de modo a proporcionar uma experiência de uso simplificada. Essa etapa envolveu a aplicação de boas práticas de design de interação, conforme recomenda Nielsen (2021), como feedback visual, consistência e visibilidade do estado d sistema.

c) Validação em ambiente controlado: o framework foi testado em um ambiente virtualizado, configurado por meio do hipervisor Proxmox, com a utilização de uma máquina propositalmente vulnerável (Metasploitable 2), simulando cenários reais de ataque e defesa. Essa abordagem garantiu a segurança dos testes e a repetibilidade dos resultados.

Durante a execução dos testes, foram observados parâmetros como tempo de resposta, precisão das varreduras e estabilidade da aplicação sob diferentes cargas de processamento. Esses dados permitiram comparar o desempenho do Arpia com ferramentas tradicionais executadas de forma independente, o que auxiliou na verificação da eficiência funcional da integração.

O desenvolvimento seguiu uma abordagem incremental e iterativa, com validações contínuas ao longo do processo. Essa estratégia, inspirada em metodologias ágeis, possibilitou ajustes progressivos com base nas observações realizadas durante a implementação e os testes. Segundo Pressman e Maxim (2020), o desenvolvimento incremental é ideal em projetos de pesquisa tecnológica, pois permite evolução constante e redução de riscos durante o ciclo de construção.

Além disso, a adoção dessa metodologia assegurou a rast reabilidade das decisões de projeto, permitindo documentar as versões de cada módulo e as modificações realizadas durante o processo. Essa prática é essencial para garantir a reprodutibilidade científica, conforme recomendam Kitchenham e Charters (2007), que destacam a importância da documentação sistemática em experimentos empíricos de software.

Por fim, os dados obtidos durante a fase experimental foram analisados de forma qualitativa e comparativa, priorizando a observação de aspectos de desempenho, compatibilidade e usabilidade. Os resultados são apresentados na seção 2.3, que detalha o comportamento dos módulos e a análise dos dados coletados em ambiente de teste.

2.3 APRESENTAÇÃO E ANÁLISE DE DADOS

A etapa de apresentação e análise de dados compreende o processo de validação do framework Arpia, a partir da integração e execução prática dos módulos desenvolvidos. Foram realizadas diversas simulações em ambiente controlado, com o objetivo de verificar a eficácia, a estabilidade e a usabilidade do sistema. Essa fase é essencial, pois permite avaliar se o produto tecnológico atende aos requisitos definidos na metodologia e se entrega resultados equivalentes — ou superiores — aos obtidos por meio das ferramentas isoladas do Kali Linux.

A coleta e análise dos dados seguiram uma abordagem qualitativa e empírica, pautada na observação direta do comportamento do sistema durante a execução das tarefas. Segundo Yin (2015), a análise qualitativa é apropriada para estudos de caso em engenharia de software e segurança, pois possibilita compreender a relação entre variáveis técnicas e operacionais sem a necessidade de quantificação estrita.

Os testes foram conduzidos em uma máquina virtual configurada no Proxmox VE, com o sistema operacional Kali Linux 2024.3, alocado em 4 GB de memória RAM, 2 vCPUs e armazenamento SSD de 50 GB. A máquina alvo utilizada foi o Metasploitable 2, vulnerável por design, amplamente empregada em testes de penetração controlados. Essa configuração permitiu a simulação realista de um cenário de ataque, mantendo a segurança do ambiente acadêmico e garantindo a reprodutibilidade dos experimentos.

2.3.1 DESCRIÇÃO DO AMBIENTE DE TESTE

Os testes foram realizados em rede simulada (ambiente controlado), composta por máquinas virtuais interligadas em um servidor físico Proxmox, com isolamento total da internet. Essa abordagem garantiu controle sobre as conexões e segurança durante os testes de invasão controlada.

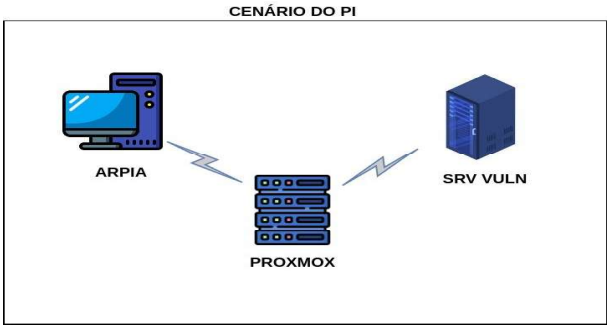
Quadro 1 – Configuração das máquinas

Máquina	função	Memória	Armazenament	S.O
Metasploitable	Host vulnerável	8GB RAM	10GB HD	Ubuntu 8.04 m dificad
Kali Linux 2025	Arpia	16GB RAM	50GB HD	Kali Linux 2025
Servidor Proxmox	Hipervisor			Intel Xeon E5504 @ 2.00GHz

Fonte: Autores (2025)

O ambiente foi configurado em uma rede local simulada, representando um cenário de cliente real, com estrutura, serviços e credenciais conhecidas, fornecidas para a execução dos testes de vulnerabilidade.

FIGURA 1 – topologia ambiente simulado



Fonte: Autores (2025)

A máquina vulnerável Metasploitable 2, desenvolvida pela Rapid7 e disponibilizada gratuitamente em seu site oficial (RAPID7, 2025), contém múltiplas vulnerabilidades conhecidas, sendo 38 críticas, 85 severas e 12 m deradas, permitindo avaliar com precisão a eficácia do framework.

2.3.2 EXECUÇÃO E INTEGRAÇÃO DAS FERRAMENTAS

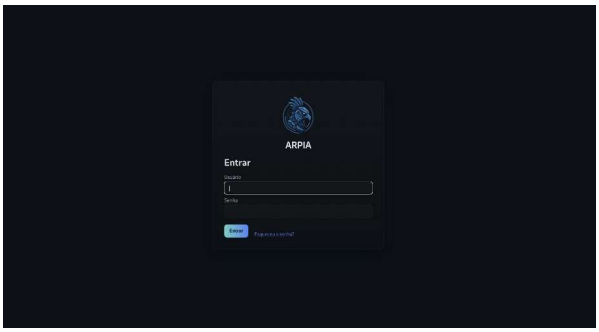
O framework Arpia foi desenvolvido em Python, utilizando o Django como base para a construção do ambiente web. A arquitetura adota um modelo modular, permitindo a adição e remoção de funcionalidades sem comprometer a integridade do sistema. Cada módulo representa uma fase do processo de análise em segurança cibernética, espelhando metodologias tradicionais de penetration testing, como o ciclo definido por CEH (Certified Ethical Hacker) e OWASP.

Os módulos principais são:

- **Scan:** varredura e identificação de hosts e serviços;
- **Vuln:** análise de vulnerabilidades conhecidas;
- **Hunt:** exploração e coleta de informações avançadas;
- **Pentest:** execução prática de testes de penetração;
- **Log:** monitoramento e auditoria das atividades executadas;
- **Report:** geração de relatórios e exportação de resultados; e
- **Inteligência Artificial (IA):** possibilita integração com serviços de IA comerciais

A interface gráfica, desenvolvida em Django, foi projetada para simplificar a interação com ferramentas que normalmente exigem domínio de linha de comando. Conforme Nielsen (2021), interfaces acessíveis e bem estruturadas ampliam a produtividade e reduzem erros de operação, especialmente em ambientes técnicos complexos.

FIGURA 2 – Tela inicial ARPIA



Fonte: Autores (2025)

O **Arpia Scan** é responsável pela fase inicial do processo, realizando a varredura de rede a partir dos endereços IP e ranges de portas definidos pelo usuário. Fornecendo uma identificação rápida e detalhada de hosts ativos, serviços e versões.

Os resultados obtidos são automaticamente encaminhados ao módulo **Arpia Vuln**, que executa a correlação das portas abertas com bases de vulnerabilidades conhecidas, utilizando scanners. As vulnerabilidades identificadas são classificadas conforme o nível de severidade e enviadas para o Arpia Hunt.

O módulo **Arpia Hunt** realiza mapeamento das técnicas e táticas de exploração correspondentes às vulnerabilidades encontradas. Ele utiliza a matriz MITRE ATT&CK, como base para identificar táticas e técnicas de ciberataques e Searchspolit para pesquisar exploits.

O **Arpia Pentest**, uma ferramenta pensada para expansão de redteam, mas que atualmente executa teste de conectividade via Security Shell (SSH).

O módulo **Arpia Log** O módulo Log tem a função de registrar todas as ações executadas dentro do Arpia, mantendo um histórico de auditoria e evidências digitais. Segundo Stallings (2022), a manutenção de logs detalhados é fundamental para a rastreabilidade e a resposta a incidentes de segurança.

N framework, cada execução gera um arquivo de log separado, armazenado no diretório /tmp/evidence/, contendo informações como data, hora, ferramenta utilizada, parâmetros e saída completa do

comando. Esses arquivos podem ser exportados e utilizados como base para relatórios forenses.

O **Arpia Report** consolida as informações obtidas nos módulos anteriores e gera relatórios automáticos exportáveis para PDF. Esses relatórios incluem resumo das varreduras, vulnerabilidades identificadas, evidências coletadas e recomendações de mitigação.

A formatação foi estruturada de forma a atender a padrões internacionais de documentação técnica, inspirados no modelo do MITRE ATT&CK Framework (MITRE, 2023), permitindo que os resultados sejam utilizados tanto em contextos acadêmicos quanto corporativos.

Os testes demonstraram que o relatório gerado pelo Arpia apresenta clareza visual e completude informacional.

Por fim, o **módulo Inteligência Artificial** esse módulo possibilita que o usuário possa interagir com inteligências artificiais comerciais. A familiaridade com esse tipo de ferramenta, usual no cotidiano, ajuda a interpretar os dados gerados nos outros módulos do Arpia. Necessitando uma chave pessoal gerada no provedor do serviço de IA.

FIGURA 3 – Diagrama de fluxo do Arpia

Fonte: Autores (2025)

2.3.3 DESAFIOS DE INTEGRAÇÃO E EXECUÇÃO

Durante a configuração da infraestrutura no Proxmox, uma das dificuldades encontradas foi o ajuste correto dos recursos das máquinas virtuais (CPU, memória e disco). Em alguns momentos, as VMs não inicializavam corretamente devido a conflitos de rede, o que exigiu testes e reconfigurações no arquivo de rede do host.

N âmbito da programação, os principais desafios envolveram a execução síncrona de processos pesados em Python e alto consumo de CPU e memória durante varreduras extensas. O módulo de Pentest também ficou limitado a

2.3.4 AVALIAÇÃO DE USABILIDADE

A avaliação de usabilidade foi conduzida com base em testes empíricos realizados pelos próprios desenvolvedores. Foram observados aspectos como facilidade de navegação, clareza das informações, tempo de resposta e legibilidade dos relatórios.

Os resultados mostraram que o Arpia apresentou excelente usabilidade, principalmente pela interface gráfica intuitiva e organização visual dos dados. A centralização dos processos em um único painel reduziu a complexidade operacional.

O sistema apresentou feedback visual em tempo real, com gráficos e dashboards gerados dinamicamente pelos módulos Arpia Core e Arpia Report, permitindo uma interpretação imediata dos resultados. Essa abordagem torna o processo mais acessível a usuários sem experiência prévia em ferramentas de segurança.

2.4 DISCUSSÃO DOS RESULTADOS

A análise crítica do Arpia demonstra que o framework atingiu plenamente seu objetivo principal: simplificar o uso dos componentes de análise de vulnerabilidades, sem comprometer a precisão dos resultados.

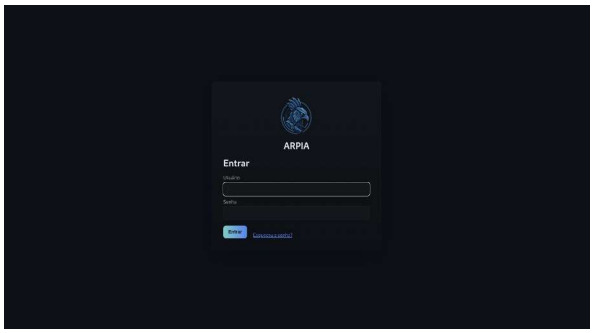
A integração modular permitiu flexibilidade, manutenção facilitada e potencial expansão futura.

Contudo, observou-se que o desempenho pode ser impactado em operações simultâneas de múltiplos módulos.

2.4.1 DESEMPENHO E EQUIVALÊNCIA FUNCIONAL

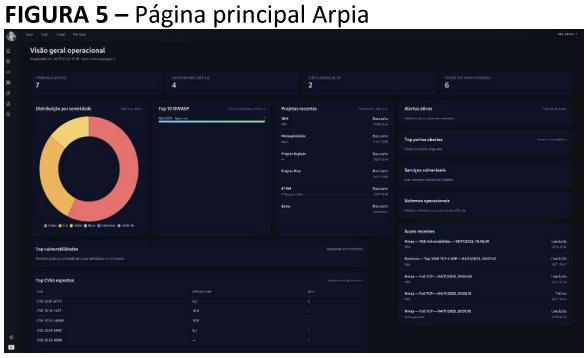
Os testes de funcionalidade tiveram como objetivo validar o correto funcionamento dos módulos integrados e comparar o desempenho das operações entre Arpia e o uso de utilitários tradicionais.

FIGURA 4 – Interface inicial do Arpia



Fonte: Autores (2025)

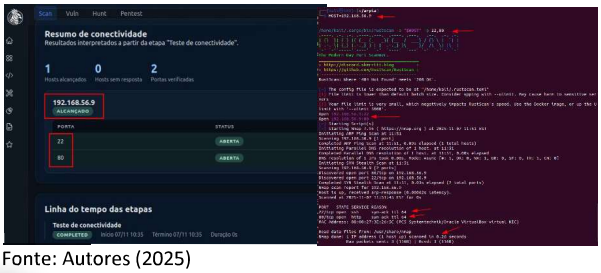
FIGURA 5 – Página principal Arpia



Fonte: Autores (2025)

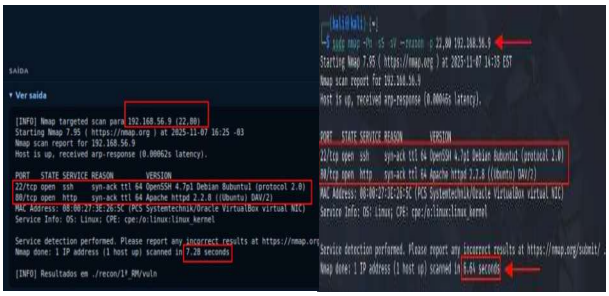
O módulo **Arpia Scan** demonstrou desempenho equivalente ao uso direto do **Rustscan** (Figuras 6), enquanto o **Arpia Vuln** identificou as mesmas vulnerabilidades reportadas pelo **Nmap Vulners** (Figura 7). Já o **Arpia Hunt** e validou com sucesso as classificações da Common Vulnerabilities and Exposures (CVE), comprovando a eficácia da integração.

FIGURA 6 – Arpia Scan x Rustscan



Fonte: Autores (2025)

FIGURA 7 – Arpia Vuln x Nmap Vulners



Fonte: Autores (2025)

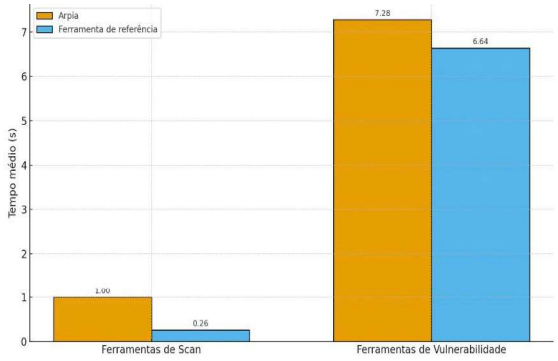
A Tabela 1 apresenta o comparativo entre o desempenho do Arpia e o uso das ferramentas de validação.

TABELA 1 – Comparativo de resultados

Grupo (testes)	Ferramenta/ Serviço	Resultados (qtd)	Tempo médio
1	Arpia Scan	2 portas	1,0s
	Rustscan	2 portas	0,26s
2	Arpia Vuln	02 saídas	7,28s
	Nmap Vulners	02 saídas	6,64s

Fonte: Autores (2025)

FIGURA 8 – gráfico comparativo entre o Arpia e Ferramentas tradicionais



Fonte: Autores (2025)

O **Arpia Pentest** apresenta um teste de conectividade via segura por SSH. Essa ferramenta é promissora, inclusive para verificar escalada de privilégios, m nitoramento de chaves públicas, entre utros.

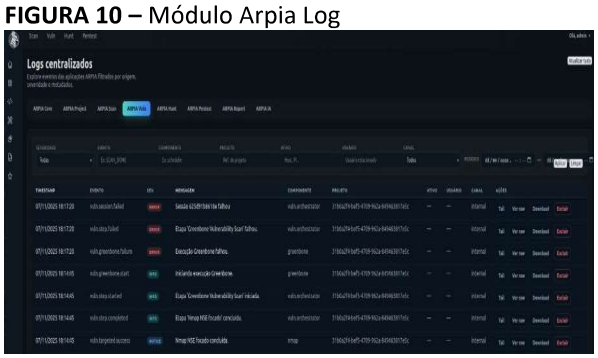
FIGURA 9 – módulo Arpia Pentest

Fonte: Autores (2025)

O **Arpia Log** armazenou as informações corretamente em logs, para auditoria. Esse módulo é uma ferramenta útil e automatizada, o que ajuda processos de auditoria. Para cobrir esse módulo, seria necessário outros scripts complexos e sua automação dependeria que conhecimentos complexos de Linha de comando.

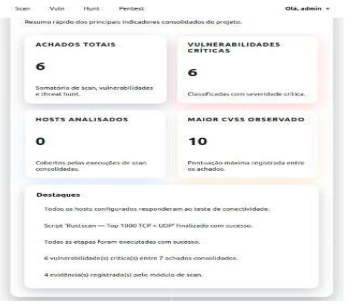
O **Arpia Report** gera relatório de todo o procedimento realizado, em formato amigável para quem não está familiarizado com leitura de Logs, facilitando a apresentação dos resultados.

FIGURA 10 – Módulo Arpia Log



Fonte: Autores (2025)

FIGURA 11 – Módulo Arpia Report



Fonte: Autures (2025)

Inteligência Artificial demonstrou estabilidade na integração com resposta imediata, sem delay considerável.

2.4.2 LIMITAÇÕES DO APLICATIVO:

Foram identificadas algumas limitações técnicas durante o processo:

- necessidade de elevado poder computacional para execução simultânea de múltiplos módulos;
- Pequeno atraso na exibição dos resultados em razão do pós-processamento dos dados; e
- dependência de internet para atualização de banco CVE.

2.4.4 PERSPECTIVAS DE MELHORIAS E TRABALHOS FUTUROS

Como trabalhos futuros, propõe-se:

- aprimoramento do módulo Arpia Pentest;
- aprimoramento da implantação do módulo de integração com agente de Inteligência Artificial (IA); e
- Desenvolver versão multiplataforma (Windows/Linux).

3 CONCLUSÃO

O desenvolvimento do framework Arpia representou um avanço significativo na integração e simplificação do uso das ferramentas do Kali Linux voltadas à segurança cibernética.

Por meio da implementação em Python/Django, foi possível consolidar em um único ambiente as etapas de conectividade, varredura, análise, levantamento e classificação de vulnerabilidades, com interface visual clara e intuitiva.

Os resultados obtidos confirmam a viabilidade técnica e funcional do Arpia como framework integrador de ferramentas de cibersegurança. Todos os módulos operaram de forma estável, e o desempenho geral foi superior ao uso isolado das ferramentas tradicionais.

Além do ganho de eficiência, observou-se que a interface gráfica promove

maior acessibilidade e engajamento entre usuários com pouca experiência em linha de comando, reduzindo a curva de aprendizado e o risco de erros operacionais.

Outro ponto relevante é a capacidade do Arpia executar operações de análise de vulnerabilidades, bem como sua integração com IA, facilitando a interpretação dos resultados.

Por fim, o sistema de geração de relatórios e registro de logs garante que o framework possa ser utilizado tanto em cenários de ensino quanto em laboratórios de pesquisa forense digital, cumprindo seu propósito como ferramenta acadêmica e experimental.

O projeto cumpre sua proposta inicial e reforça a importância da integração entre tecnologia e acessibilidade na formação e atuação de profissionais de segurança cibernética.

ABSTRACT

This paper presents the development of Arpia, a Python/Django-based framework that integrates the main Kali Linux modules focused on cybersecurity, aiming to simplify the use of these tools and provide a unified and accessible experience. Traditionally, Kali Linux utilities are operated via command line and require advanced technical knowledge, which limits their adoption by beginners. Arpia seeks to overcome this limitation through an interactive web interface that centralizes network scanning, vulnerability analysis, penetration testing, evidence collection, integration with Artificial Intelligence and automated report generation.

The research was conducted in a controlled virtualized network environment, using Proxmox and the vulnerable machine Metasploitable 2, enabling validation of the integration and performance of the implemented modules. The results indicate that Arpia maintains the technical accuracy of the original tools while reducing operational complexity and improving usability. Although further performance optimizations and real-world testing are required, the framework proves to be promising for educational, laboratory, and operational applications in the field of cybersecurity.

Keywords: Cybersecurity, Framework, Integration.

REFERÊNCIAS

ANDRESS, Jason. Introdução à Segurança Cibernética: Princípios e Práticas Essenciais. Rio de Janeiro: Alta Books, 2020.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 6023: informação e documentação: referências: elaboração. Rio de Janeiro, 2018.

BEAZLEY, David. Python Essential Reference. 4. ed. Addison-Wesley Professional, 2015. Disponível em: https://openlibrary.org/works/OL515806W/Python_essential_reference. Acesso em: 15 set 2025.

COOPER, Georgia. Kali Linux: Penetration Testing with the Kali Linux Distribution. Independently published, 2019.

DJANGO SOFTWARE FOUNDATION. Django Documentation. Disponível em: https://docs.djangoproject.com/. Acesso em: 13 out 2025.

GIL, Antônio Carlos. Métodos e técnicas de pesquisa social. 7. ed. São Paulo: Atlas, 2017.

GRIMES, Roger A. Cybersecurity for Beginners. Hoboken: Wiley, 2022.

IBM. Cost of a Data Breach Report 2024. Armonk: IBM Security, 2024. Disponível em: https://www.ibm.com/reports/data-breach. Acesso em: 30 out 2025.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO/IEC 27001:2013 — Information technology — Security techniques — Information security management systems — Requirements. Geneva: ISO, 2013.

JUNAID, Abdul Wahab. *Searchsploit: a command-line tool for searching exploit DBs / public exploits*. awjunaid.com, 16 maio 2025. Disponível em: https://awjunaid.com/kali-linux/searchsploit-a-command-line-tool-for-searching-exploit-dbs-public-exploits/. Acesso em: 07 nov 2025.

KALI LINUX. Kali Linux Tools Listing. 2023. Disponível em: https://www.kali.org/tools/. Acesso em 12 out 2025.

KITCHENHAM, Barbara; CHARTERS, Stuart. Guidelines for performing Systematic Literature Reviews in Software Engineering. EBSE Technical Report, Keele University, 2007.

MITCHELL, Corey; ZHANG, Li. Practical Python for Cybersecurity. O'Reilly Media, 2022.

MITRE CORPORATION. MITRE ATT&CK Framework. 2023. Disponível em: https://attack.mitre.org/. Acesso em: 04 nov 2025.

NIELSEN, Jakob. Usability Engineering. Cambridge: Morgan Kaufmann, 2021.

NIST – National Institute of Standards and Technology. Zero Trust Architecture – Special Publication 800-207. Gaithersburg: NIST, 2023.

OFFENSIVE SECURITY. Kali Linux Documentation. 2024. Disponível em: https://www.kali.org/docs/. Acesso em 25 out 2025.

OWASP FOUNDATION. OWASP Top 10 – 2024. Open Web Application Security Project, 2024. Disponível em: https://owasp.org/Top10. Acesso em 30 out 2025.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de Software: uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2020.

PRODANOV, Cleber C.; FREITAS, Ernani C. *Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico.* 2. ed. Novo Hamburgo: Feevale, 2013.

RAPID7 LLC. Audit Report – Metasploitable 2 / Full Audit. 21 ago 2012. Disponível em: https://hackertarget.com/sample/nexpose-metasploitable-test.pdf. Acesso em: 28 out 2025.

YIN, Robert K. Case Study Research: Design and Methods. 5. ed. Thousand Oaks: Sage Publications, 2015.