RedKit: A Lightweight Penetration Testing Framework using Docker-based Isolation

Mohammed Abdelfattah, PhD
Modern Academy for Computer Science
Cairo, Egypt

Youssef Hamdy Abdelazeem
Modern Academy for Computer Science
Cairo, Egypt

Mohamed Mahmoud Hanafi
Modern Academy for Computer Science
Cairo, Egypt

Ziad Mahmoud Mohamed
Modern Academy for Computer Science
Cairo, Egypt

Abdallah Waleed Abdelmajeed
Modern Academy for Computer Science
Cairo, Egypt

Ahmed Mamdouh Salem
Modern Academy for Computer Science
Cairo, Egypt

ABSTRACT
RedKit is a Security as a Service (SECaaS) platform designed to facilitate penetration testing across the entire life cycle through a microservices architecture. RedKit includes both manual and automated testing. In manual testing, RedKit offers custom-built Docker containers as a pre-configured environment that includes a web proxy ready for testing. In automated testing, RedKit features an AI-driven vulnerability scanner to automate repetitive tests, reducing the effort required of penetration testers. RedKit includes information gathering, reconnaissance tools, and AI report generation. RedKit integrates all these features into a cloud-based, all-in-one framework, a low-effort solution for end-to-end security assessments. By integrating Docker and merging automated testing with manual testing, RedKit builds a full penetration testing framework with minimal resource overhead. By accomplishing 60% of resource management and 90% of time saving for setting up the environment.

General Terms
Cyber Security, Penetration Testing.

Keywords
Containerization, Penetration Testing, Web Proxy, SECaaS (Security as a Service), Methodology, LLM, Reporting, Payloads, Ethical Hacking, Docker.

1. INTRODUCTION
In the modern world, websites and web services have been an important backbone of our society, making cybersecurity a crucial component to keep the web infrastructure safe from potential threats. As cybersecurity threats become more complex, penetration testing—simulating potential unwanted attacks to exploit the system—has become a more in-demand service than before. According to the European Union Agency for Cybersecurity (ENISA), 21% of the attacking tactics and techniques used by unwanted parties involved the exploitation of vulnerabilities [1]. Organizations must be keen to test their web services to identify vulnerabilities before unethical hackers exploit them. This protects the organization's systems and sensitive data while maintaining client trust.

The traditional approach to penetration testing often faces many challenges. Most penetration testers rely on heavy Virtual Machines (VMs) for testing, resulting in significant waste of hardware resources such as RAM and CPU [2]. Also, the manual setup of these environments is not easy for a new penetration tester. Testers frequently spend time configuring proxy settings and other tools and troubleshooting network isolation before a single test can be performed. This repetitive, manual setup wastes time and reduces penetration testers' productivity.

In previous research and on current platforms, researchers have tried to address these challenges by providing testing methodologies or relying on automated testing, but a significant gap remains in deployment and resource management. Most of these solutions still rely on virtualization, which is not resource-efficient. Other solutions don't provide a fresh, ready environment that bridges the gap between manual and automated testing, leading to a lack of integration between them. which complicate the testing life cycle and affect penetration testers' productivity and efficiency.

This paper presents several contributions to the field of penetration testing. RedKit is a smart Security as a Service (SECaaS) that offers an innovative approach to penetration testing by eliminating Virtual Machines (VMs) and their configurations. Using a lightweight, isolated microservices architecture, RedKit provides a pre-configured environment that is more efficient and convenient than traditional setups. The main objective of this paper is to explain how a cloud-ready framework moved away from manual configuration while retaining the ability to integrate with other services. Furthermore, the paper will explain why containerization is more efficient than traditional VMs.

2. BACKGROUND AND RELATED WORK
Penetration testing is the process of performing security tests on a specific target in a network or web application. Testing's main job is to identify vulnerabilities in the target before any unauthorized party can exploit them, and then mitigate them immediately.

Penetration testing can be done using two approaches: manual and automated testing. Each has its own advantages and disadvantages. In automated testing, it is faster than manual testing,

better at detecting direct common vulnerabilities, but it struggles with logic vulnerabilities, where manual testing steps in. In manual testing, the penetration tester tests the vulnerability by themselves. This allows penetration testers to test more complex scenarios to evaluate the environment and ensure it is well-secured. Usually, it takes more time than automated testing.

2.1 Penterep

A complete platform that combines the best of manual and automated testing. Its unique features make it work as efficiently and effectively as possible. It facilitates the penetration testing process by requiring less time and effort than other options. However, the platform does not use containerized environments based on Docker, which may limit deployment flexibility and lightweight service isolation. In addition, Penterep does not provide integrated proxy-based traffic interception capabilities for real-time request manipulation and analysis during web application testing [3], [4].

2.2 BurpSuite

A group of integrated tools used for testing web applications that runs locally on your device. BurpSuite aims to be an all-in-one set of tools, and its capabilities can be enhanced by installing add-ons that are called BApps. It is the most popular tool among professional web app security researchers and bug bounty hunters. Burp Suite is available as a free community edition and a paid professional edition. Burp Suite primarily focuses on web application testing and does not provide integrated external reconnaissance capabilities or report generation. In addition, local deployment may result in increased memory and resource consumption, particularly during large-scale testing sessions or when multiple extensions are enabled [5].

2.3 OWASP ZAP

An open source penetration testing tool, formerly known as OWASP ZAP. It is a multidimensional tool often used by penetration testers, bug bounty hunters, and developers to scan web applications for security risks during application testing. ZAP offers many features, including active and passive scanning and API testing capabilities. ZAP primarily focuses on application-layer testing and lacks integrated external reconnaissance and advanced automated reporting capabilities. Furthermore, running the platform locally can result in significant resource utilization during extensive scanning operations. In addition, ZAP may produce false-positive findings during automated vulnerability analysis, requiring manual verification [6].

2.4 w4af

A complete environment for security analysis and attack simulation of web-based services. w4af relies on modular plugins for the discovery, analysis, and validation of web applications, enabling customization of the penetration testing methodology. It has several limitations. The platform may generate high false positive results and experience performance instability or crashes during intensive scanning operations. In addition, w4af has limited compatibility with modern single-page applications (SPA) and JavaScript-heavy web environments. The framework also lacks comprehensive external reconnaissance capabilities, such as subdomain enumeration and DNS analysis [7].

3. METHODOLOGY

Penetration testing is always performed in line with proven, industry-recognized methodologies that evolve alongside the latest cutting-edge techniques [8], [9]. There are 5 key stages to most penetration tests based on common industry methodology frameworks[8].

3.1 Scoping

Before performing the penetration test, the tester will collaborate with the client to define an effective and appropriate scope that meets the client's needs and goals. for example, IP addresses, web applications, and internal systems.

3.2 Reconnaissance (Information Gathering)

Simulating the approach of a malicious actor, penetration testers gather as much information as possible about the client environment. The reconnaissance phase is divided into passive and active components:

3.2.1 Passive Reconnaissance

RedKit uses WHOIS lookup to collect domain information. The framework also includes custom subdomain enumeration.

3.2.2 Active Reconnaissance

RedKit contains custom endpoint fuzzing to discover sensitive paths and hidden endpoints. The framework also integrates Nmap for port and service discovery.

3.3 Vulnerability Analysis

Penetration testers will identify the vulnerabilities that threaten the security of the client organization’s environment and develop an exploitation strategy, both manually and using the latest automated tools. RedKit includes a lightweight vulnerability scanner integrated with a large language model (LLM) to analyze discovered endpoints and detect issues such as SQL injection.

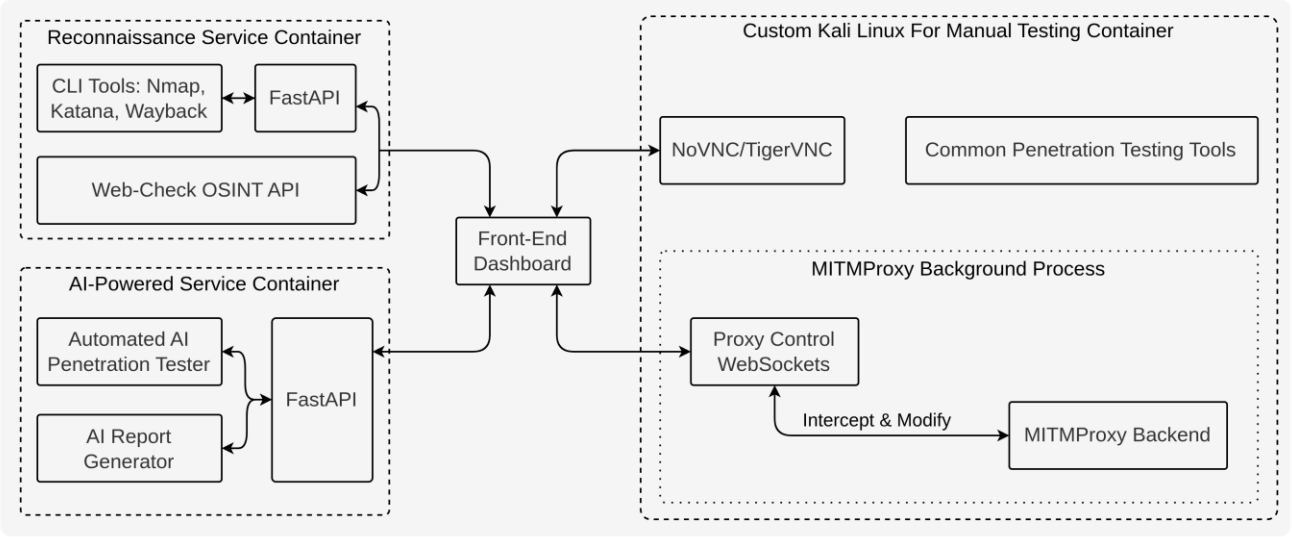

Figure 1: High-level implementation architecture of RedKit.

3.4 Exploitation and Post-Exploitation

The tester will attempt to exploit the vulnerabilities they identified in the previous stage to determine how a genuine threat actor could impact the client organization. RedKit implements a custom proxy running in an isolated container and connects it to the external framework interface. This enables the interception of HTTP traffic before forwarding, which is very important for validating vulnerabilities that require manual manipulation of the request and response.

3.5 Reporting and AI Assistance

Following completion of the test, the pen tester will detail their findings in a report and provide a debrief meeting at the end of the assessment to help the client navigate the remediation process. RedKit provides a reporting template and chatbot :

3.5.1 AI-based Report Generation

automatically generates a full penetration testing report from collected findings.

3.5.2 Chatbot Assistant

supports users with vulnerability explanations (e.g., XSS), suggests payloads, and provides guidance on using the framework.

4. IMPLEMENTATION

As a prototype, the current version demonstrates a flexible, scalable containerized architecture that showcases RedKit's core capabilities. These are the main implementation points that show in detail how RedKit works.

4.1 Containerized Service Engine

The framework highlights the use of Docker containerization to improve portability and create an isolated environment [10]. Each service is encapsulated and isolated in a separate container, and all services communicate over a shared network.

4.1.1 Isolation

Isolating tools and services from the host system mitigates dependency conflicts and reduces potential risks during active scanning.

4.1.2 Communication

Communication between services is established via APIs and WebSockets, allowing the front end to communicate with each service independently [11], [12].

4.2 Reconnaissance

Combining passive and active scanning, RedKit gathers accurate information about the target's subdomains and endpoints. Reconnaissance is performed through the following workflow.

4.2.1 Target general information

RedKit relies on WHOIS CLI and Web-Check API to collect DNS and registration information, in addition to target technology details [13], [14].

4.2.2 Subdomain reconnaissance

In active reconnaissance, RedKit uses the DNS. resolver Python library to resolve and validate subdomains [15]. For passive reconnaissance, the framework uses crt.sh certificate transparency logs [16].

4.2.3 Port scanning

RedKit performs active scanning using Nmap on commonly used ports for discovered subdomains [17].

4.2.4 Endpoint discovery

RedKit performs endpoint fuzzing for active discovery and uses Waybackurls and Gospider for passive historical URL extraction from Internet Archive sources [18]–[20].

4.2.5 Real-time streaming

To handle long-running scans, RedKit uses WebSockets for responsive communication between backend services and the front end [12].

4.3 User Custom Docker Container

The main purpose of this paper and the novel contributions of this paper are to introduce a modern approach to manual penetration testing that achieves higher efficiency than existing methods, as shown in the next paragraph.

4.3.1 Custom containerized environment

RedKit provides an isolated Kali Linux Docker container with essential penetration-testing tools for manual testing [21], [22].

4.3.2 Remote browser-based access

RedKit uses TigerVNC to capture the Kali desktop and noVNC to expose it through the browser, enabling cloud-based testing

access [23], [24].

4.3.3 Custom containerized environment

Mitmproxy runs automatically inside the container, SSL certificates are imported, and the browser is preconfigured for immediate proxy usage [25].

4.3.4 Remote browser-based access

A FastAPI WebSockets backend is integrated with the proxy to control traffic and stream outputs to the external interface [11], [12].

4.3.5 AI & third-party integration

AI services can process Mitmproxy traffic for automated scans, and integration with third-party services and MCP components is supported [25], [26].

4.4 AI-Service

To connect scan results and proxy traffic with LLMs, we built the AI services using FastAPI to ensure smooth communication between the framework and external models. The AI vulnerability scanner is not the main focus of this project; it's just a feature that supports all aspects of the penetration testing methodology [11].

4.4.1 Execution flow

Once a reconnaissance scan or intercepted request is completed, the system sends JSON-formatted results directly to the AI services to initiate automated testing.

4.4.2 Intelligence layer

This layer uses Gemini and Cohere APIs to perform vulnerability validation by analyzing post-exploit responses [27], [28].

5. TESTING APPROACH

This section presents a rigorous experimental assessment of the RedKit framework across performance and efficiency metrics, including reconnaissance accuracy, resource management, and startup efficiency. The evaluation was designed to measure whether the presented container-based architecture offers improvements over the current environments. All experiments were performed under the same controlled parameters to ensure statistical validity.

5.1 Evaluation Metrics

The following key performance indicators (KPIs) were defined to measure the efficiency of the framework with other known environments:

5.1.1 Startup Time

Startup Time is measured from environment initialization to a fully functional web proxy, with the environment ready for the tester to begin engaging with the target.

5.1.2 Memory Consumption

Memory Consumption is measured as both host-allocated and guest-reported RAM usage, which determines the scalability of concurrent testing sessions.

5.1.3 Reconnaissance Coverage

Reconnaissance is quantified by the number of subdomains discovered, the number of open ports identified, and the number of unique endpoint URLs extracted per target domain.

These metrics together indicate the practical utility of a penetration testing framework in real-world scenarios.

5.2 Experimental Setup and Methodology

All experiments were performed on an identical hardware platform comprising an Intel Core i7 processor with 16 GB of RAM running Ubuntu 24.04 LTS. Three testing environments were established as comparative baselines:

5.2.1 Kali Linux Virtual Machine

Traditional Kali Linux virtual machine installed through Oracle VirtualBox [29] with 4 GB allocated RAM and 2 CPU cores, with all penetration testing and ethical hacking tools.

5.2.2 Kali Linux Docker Image

Kasm Workspaces Docker image [21] based on Kali Linux, with configurable resource limits of 3.4 GB. This image includes most of the well-known penetration testing tools and ethical hacking tools.

5.2.3 RedKit Custom Kali Linux Image

RedKit framework running on Docker Engine [10] with a 1.2 GB container memory limit. The image is built on top of a minimal Kali Linux Docker image, with Mitmproxy [25] already running in the background.To achieve consistent, reliable proxy functionality across all the environments. Evaluations were performed repeatedly across 5 trials under the same environmental conditions. Each measurement was repeated across five consecutive trials under identical environmental conditions to avoid temporary system slowdown or caching that may affect the results. The provided values represent arithmetic means calculated from the trial set. Startup time was measured from the moment the environment launched until the proxy began reading traffic in the browser. Memory consumption was measured using the system monitoring utility btop [30] at its highest utilization during active reconnaissance tasks.

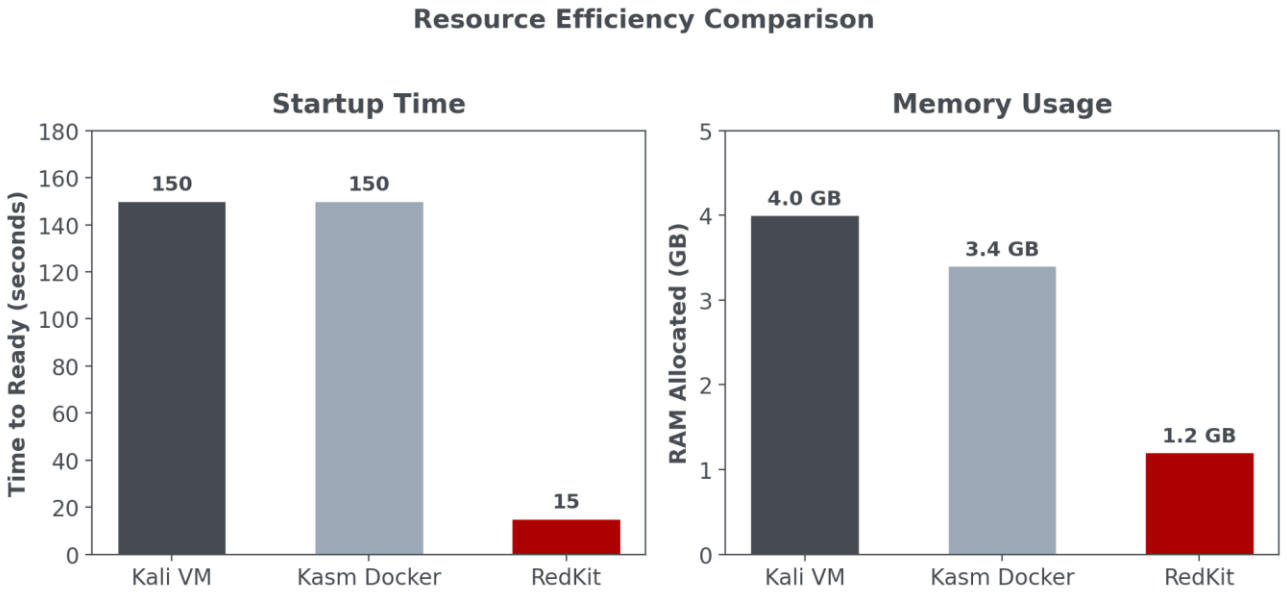

Figure 2: Resource and setup-time comparison between VM, Kasm Docker, and RedKit

5.3 Comparative Analysis and Discussion

5.3.1 Reconnaissance Performance Analysis

The reconnaissance features of RedKit were tested against five anonymized production web domains with different technology architectures. Passive reconnaissance techniques were executed first, providing DNS record queries, WHOIS lookups [13], and certificate transparency logs via crt.sh [16], as well as historical URLs from the Internet Archive [19]. Active reconnaissance was performed using Nmap service scanning [17] and custom endpoint fuzzing with common web wordlists.

The automated subdomain enumeration procedure yielded a mean discovery rate of 37.8 subdomains per target, with approximately 91% true positives. The automated approach identified several subdomains that manual reconnaissance had missed, especially testing environments and legacy services that were no longer recognized by active DNS records but remained resolvable. Historical URL retrieval returned an average of 823 unique endpoint URLs per domain, providing a detailed attack target map for later vulnerability analysis. The endpoint discovery phase achieved complete enumeration of visible paths within 10 minutes per target, well beyond manual browsing, which typically requires 45 to 60 minutes for equivalent coverage.

Table 1 summarizes the reconnaissance results across all five target domains. This architectural advantage is all thanks to RedKit's parallel execution model in its containerized engine running on Docker. Unlike standalone tools that handle reconnaissance phases sequentially, the microservices architecture allows passive information gathering, active scanning, and data normalization to run simultaneously across independent containers, eliminating the sequential bottlenecks that plague traditional approaches.

Table 1: Reconnaissance Results Across Five Anonymized Target Domains

Target Domain	Subdomains Discovered	Open Ports Identified	Unique URLs Extracted
Web App 1	50	3	1,042
Web App 2	35	2	245
Web App 3	44	4	689
Web App 4	34	6	812
Web App 5	26	3	1,327
Mean Value	37.8	3.6	823

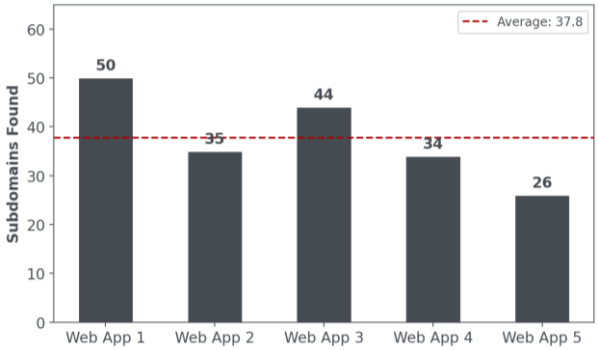

Figure 3: Reconnaissance results across five anonymized web application targets

5.3.2 Resource Efficiency and Startup Performance

The technical efficiency of a penetration testing environment is determined by two factors: the time required to reach a test-ready state and the memory allocated during active use. Both metrics directly affect tester productivity and the practical success of running parallel testing sessions.

Figure 2 shows a noticeable gap in startup performance across the three evaluated environments. The RedKit framework reached full functionality in a mean time of 15 seconds (s.d. = 2.1 s across five trials), which represents a 90% reduction

compared to both the Kali Linux virtual machine and the Kasm Docker image [21], each requiring approximately 150 seconds to become ready. This improvement is structurally possible due to the elimination of two significant consumption sources: first, the virtual machine must execute a complete operating system boot sequence, including kernel initialization, service activation, and display manager startup, all before BurpSuite [5] configuration with the browser; second, the Kasm Workspaces Docker Image [21], while container-based, initializes a full desktop environment with a window manager and VNC server, all before BurpSuite [5] configuration with the browser. RedKit's custom container bypasses both of these by pre-configuring the Mitmproxy [25] background service, auto-importing SSL certificates, and setting browser proxy settings during the image build process, reducing runtime initialization to just service startup.

Figure 2 shows that memory consumption analysis reveals similarly significant advantages. The Kali virtual machine required 4.0 GB of host-allocated RAM and reported 2.8 GB of actual in-guest usage at peak load during active reconnaissance. The Kasm Docker image [21] consumed 3.4 GB of allocated space and 2.1 GB of actual space. In the opposite situation, RedKit used only 1.2 GB of allocated memory and 1.0 GB of actual memory, achieving a 60% reduction in host memory demand relative to the VM baseline and a 15% improvement over Kasm [21]. The main design rationale lies in the difference between hypervisor-based virtualization and operating-system-level containerization.

Virtual machines require a separate guest kernel to operate and emulate actual hardware devices, each causing memory waste that increases regardless of load intensity. Containers share the host kernel and execute as isolated process groups, eliminating hypervisor high memory usage entirely. While RedKit's purpose-built container provides a graphical desktop environment for user interaction, it optimizes memory efficiency by eliminating non-essential system services and background processes. The primary architectural advantage lies in using a native, lightweight Mitmproxy [25] instance running in the background, rather than relying on resource-heavy alternatives like BurpSuite [5], which increases the memory consumption of both the VM and the Kasm Docker Image [21].

Table 2: Comparative Analysis of Resource Consumption and Startup Latency Across Testing Environments

Performance Metric	Virtual Machine	Kasm Workspaces	RedKit Custom Image
Startup (seconds)	150	150	15
Latency Improvement	--	--	90%
Host-Allocated RAM (GB)	4.0	3.4	1.2
Guest-Reported RAM (GB)	2.8	2.1	1.0
RAM Efficiency vs. VM	--	15%	60%

6. LIMITATIONS

Despite documented improvements in resource efficiency and reconnaissance performance, the current proof-of-concept implementation of RedKit has some limitations that must be mentioned to properly clarify the results and guide future development efforts.

6.1 AI Scanner Maturity

The integrated AI vulnerability scanner is currently in an early developmental stage and has not completed organized training across different vulnerability taxonomies. While the current implementation demonstrates the ability to detect common vulnerabilities such as SQL injection and cross-site scripting (XSS), its detection accuracy and false-positive rate have not been verified against existing benchmarks, and systematic benchmarking against standard vulnerability associations, such as the OWASP Benchmark Project, is required before production deployment.

6.2 Persistence and Session Management

The current architecture operates entirely in memory without a persistent data storage layer. Scan results, session state, and configuration parameters are lost upon container restart, limiting the framework's applicability to multi-session engagements and collaborative testing scenarios. Integration with PostgreSQL and Redis [31],[32] has been identified as a critical path item for the next development phase.

6.3 Container Security Isolation

Using Docker containers may compromise the host system via the container escape attack because Docker isolates processes using kernel namespaces and control groups [10]. This makes the usage of virtualization technologies such as Oracle VirtualBox [29] more secure than using containerization technology. The comparative resource analysis also relies on host-level monitoring via btop [30]. This risk is particularly relevant in penetration testing contexts, where the framework intentionally interacts with potentially malicious targets. Mitigation strategies, including seccomp profile restriction, AppArmor mandatory access control, and user namespace remapping, should be implemented and evaluated before deployment in production or multi-tenant environments.

6.4 Reconnaissance Toolset Limitations

The reconnaissance phase of RedKit, while highly automated, is subject to limitations inherent to its current toolset. Future iterations can mitigate this by using a wider list of external security tools to improve the assessment. There is room to update and enhance the logic of the currently integrated utilities, allowing for deeper vulnerability discovery and reduced false-positive rates during active scanning.

7. AREAS OF IMPROVEMENTS

The current proof of concept established the core framework features. RedKit is designed for scalable growth, with future layers.

7.1 Data persistence layer

Planned integration of PostgreSQL and Redis to store tester data and manage sessions [31], [32].

7.2 Infrastructure orchestration

Kubernetes-based orchestration for managing large container workloads securely at scale [33].

7.3 More cybersecurity tools

Modular integration of additional tools and MCP servers with RedKit’s architecture [26].

7.4 More AI capabilities

Adding new AI features and vulnerability testing with smart integration with penetration testing tools.

8. CONCLUSION

Penetration testing is essential for keeping web systems secure,

but the traditional approach is often slow and resource-intensive. Most testers still rely on full virtual machines that take a long time to boot, consume a lot of RAM, and need repetitive manual setup before any real testing can start. This overhead reduces productivity and makes it harder to run multiple assessments in parallel, especially for small teams or independent researchers.

In this work, RedKit is a lightweight, container-based framework that brings manual and automated testing together in one place. Instead of launching a heavyweight VM, RedKit spins up a preconfigured Kali Linux container with Mitmproxy already running, SSL certificates preinstalled, and browser proxy settings ready to go. The framework also adds reconnaissance tools, an AI-assisted vulnerability checker, and auto-generated reporting all accessible through a simple web interface.

RedKit experiments show that this approach makes a real difference. RedKit cut environment startup time from ~150 seconds down to about 15 seconds, and reduced host memory usage from 4 GB (typical for a VM) to just 1.2 GB. At the same time, it maintained high reconnaissance quality, discovering an average of 37 subdomains and 823 endpoints per target, with a 91% true-positive rate. These numbers confirm that moving from VMs to lightweight containers doesn't sacrifice capability, it actually speeds up the workflow and makes it more scalable.

Practically, this means security testers can now launch fresh, isolated environments on demand, run several tests at once on modest hardware, and spend less time on setup and more time on actual analysis. Looking ahead, the RedKit framework plans to add persistent storage, support for Kubernetes orchestration, and tighter integration with specialized security models to further improve detection accuracy. RedKit isn't meant to replace human expertise; it's built to remove the repetitive friction so testers can focus on what really matters: finding complex vulnerabilities and helping organizations stay secure.

9. ACKNOWLEDGMENTS

The authors would like to acknowledge their supervisors, Dr. Mohammed Abdelfattah and Prof. Mahmoud Gadallah, from Modern Academy for Computer Science and Management Technology, Cairo, Egypt, for their continuous guidance, valuable support, and encouragement throughout this research.

10. REFERENCES

[1] ENISA, “ENISA Threat Landscape 2024,” 2024. [Online]. Available:https://www.enisa.europa.eu/publications/enisa-threat-landscape-2024

[2] R. Morabito, J. Kjällman, and M. Komu, “Hypervisors vs. lightweight virtualization: A performance comparison,” in Proc. IEEE Int. Conf. on Cloud Engineering (IC2E), 2015, pp. 386–393, doi:10.1109/IC2E.2015.74.

[3] V. Lazarov, P. Seda, Z. Martinasek, and R. Kummel, “Penterep: Comprehensive penetration testing with adaptable interactive checklists,” Computers & Security, vol. 154, p. 104399, 2025. [Online]. Available:https://www.sciencedirect.com/science/article/pii/S0167404825000884

[4] S. S. Patil and S. S. Shinde, “Analysis of Penetration Testing Tools,” in Proc. Int. Conf. on Inventive Research in Computing Applications (ICIRCA), 2018. [Online]. Available:https://www.researchgate.net/publication/326077274

[5] PortSwigger, “Burp Suite: The class-leading vulnerability scanner,” [Online]. Available: https://portswigger.net/burp

[6] OWASP Foundation, “OWASP Zed Attack Proxy (ZAP),”[Online]. Available: https://www.zaproxy.org/

[7] A. Riancho, “w3af: Web Application Attack and Audit Framework,”[Online]. Available: https://github.com/andresriancho/w3af

[8] PTES, “Penetration Testing Execution Standard (PTES),”[Online]. Available: http://www.pentest-standard.org/

[9] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, “Technical Guide to Information Security Testing and Assessment (NIST SP 800-115),” NIST, 2008. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf

[10] Docker Inc., “Docker Documentation,” [Online]. Available: https://docs.docker.com/

[11] S. Ramirez, “FastAPI framework, high performance, easy to learn,”[Online]. Available:https://fastapi.tiangolo.com

[12] IETF, “RFC 6455: The WebSocket Protocol,” [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6455

[13] d'Itri, M. (2026). *whois: Intelligent WHOIS client* (Version 5.5) [Computer software]. GitHub. https://github.com/rfc1036/whois

[14] A. Sykes, “Web-Check: All-in-one OSINT tool for analyzing any website,” [Online]. Available: https://github.com/Lissy93/web-check

[15] R. Halley, “dnspython: A DNS toolkit for Python,” [Online]. Available:https://www.dnspython.org/

[16] Sectigo, “crt.sh: Certificate Search,” [Online]. Available: https://crt.sh/

[17] Nmap Project, “Nmap: Free Security Scanner, Port Scanner, & Network Exploration,” [Online]. Available: https://nmap.org

[18] Jaeles Project, “GoSpider: Fast web spider written in Go,” [Online].Available: https://github.com/jaeles-project/gospider

[19] T. Nomnom, “Waybackurls: Fetch all the URLs that the Wayback Machine knows about for a domain,” [Online]. Available: https://github.com/tomnomnom/waybackurls

[20] Internet Archive, “Wayback Machine API,” [Online]. Available: https://archive.org/web/

[21] Kasm Web, “Kali Rolling Desktop Docker Image,” Docker Hub. [On-line]. Available: https://hub.docker.com/r/kasmweb/kali-rolling-desktop

[22] Offensive Security, “Kali Linux Penetration Testing OS,” [Online].Available: https://www.kali.org/

[23] Kanaka, “noVNC: HTML5 VNC Client,” [Online]. Available: https://novnc.com/

[24] TigerVNC Project, “High-performance, platform-neutral VNC,” [On-line]. Available: https://tigervnc.org/

[25] A. Cortesi, M. Hils, and T. Kriechbaumer, “mitmproxy: A free and open source interactive HTTPS proxy,” [Online]. Available: https://mitmproxy.org/

[26] Anthropic, “Model Context Protocol Specification,” [Online]. Available:https://modelcontextprotocol.io/

[27] Google DeepMind, “Gemini: The most capable AI models,”[Online].
Available: https://deepmind.google/technologies/gemini/

[28] Cohere Inc., “Cohere LLM Documentation,” [Online]. Available: https://cohere.com/

[29] Oracle Corporation, "Oracle VM VirtualBox," [Online]. Available: https://www.virtualbox.org/

[30] Aristocratos, "btop: Resource monitor," [Online]. Available: https://github.com/aristocratos/btop

[31] PostgreSQL Global Development Group, “PostgreSQL Documentation,”[Online].
Available: https://www.postgresql.org/docs/

[32] Redis Ltd., “Redis Documentation,” [Online]. Available: https://redis.io/docs/latest/

[33] Kubernetes Authors, “Kubernetes Documentation,” [Online].Available: https://kubernetes.io/docs/